Содержание

Задача 1 **3**

Задача 2 **6**

Задача 2.1 6

Задача 2.2 11

Задача 3 **15**

Задача 4 **23**

Задача 5 **32**

Заключение **45**

Список литературы **48**

Приложение **49**

Задача 1

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: Swiss.

Объясняемая переменная: *Agriculture*.

Регрессоры: *Education*, *Examination*.

1. Оценить среднее значение, дисперсию и СКО переменных *Agriculture*, *Education* и *Examination*.

Для нахождения среднего значения, дисперсии и среднего квадратического отклонения указанных переменных воспользуемся встроенными в язык R командами (Код 1).

```r
mean(swiss$Agriculture) # Ср. арифм. Agriculture = 50.65957
var(swiss$Agriculture) # Дисперсия Agriculture = 515.7994
sd(swiss$Agriculture) # СКО Agriculture = 22.71122

mean(swiss$Education) # Ср. арифм. Education = 10.97872
var(swiss$Education) # Дисперсия Education = 92.45606
sd(swiss$Education) # СКО Education = 9.615407

mean(swiss$Examination) # Ср. арифм. Examination = 16.48936
var(swiss$Examination) # Дисперсия Examination = 63.64662
sd(swiss$Examination) # СКО Examination = 7.977883
```

Код 1. Вычисление среднего арифметического, дисперсии и СКО переменных.

2. Построить зависимости вида $y = a + bx$, где y — объясняемая переменная, x — регрессор (для каждого варианта по две зависимости).

При построении моделей воспользуемся функцией *lm* пакета *lmtest* для облегчения задачи. Построим модели $\mathit{Agriculture} = a + b(\mathit{Education})$ (Код 2) и $\mathit{Agriculture} = a + b(\mathit{Examination})$ (Код 3).

```
> library("lmtest")
> summary(lm(formula=Agriculture~Education, data=swiss))

...

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  67.2432     3.9321   17.10  < 2e-16 ***
Education    -1.5105     0.2707   -5.58  1.3e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 17.65 on 45 degrees of freedom
Multiple R-squared:  0.409,    Adjusted R-squared:  0.3959
F-statistic: 31.14 on 1 and 45 DF,  p-value: 1.305e-06
```

Код 2. Построение зависимости $Agriculture = a + b(Education)$.

```
> library("lmtest")
> summary(lm(formula=Agriculture~Examination, data=swiss))

...

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  82.8869     5.6407  14.694  < 2e-16 ***
Examination  -1.9544     0.3086  -6.334 9.95e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.7 on 45 degrees of freedom
Multiple R-squared:  0.4713,    Adjusted R-squared:  0.4596
F-statistic: 40.12 on 1 and 45 DF,  p-value: 9.952e-08
```

Код 3. Построение зависимости $Agriculture = a + b(Examination)$.

3. Оценить, насколько «хороши» полученные модели по коэффициенту детерминации R^2.

Как видно из вывода консоли вызова функции *summary* (Код 2, Код 3), R^2 первой модели равняется 0.409, R^2 второй модели — 0.4713. Это означает, что обе модели получились приемлемыми в отношении точности

для парной регрессии, однако достаточно «неточными» в общем случае, поэтому их нужно значительно улучшать, добавляя дополнительные регрессоры и вводя их функции.

4. Оценить, есть ли взаимосвязь между объясняемой переменной и объясняющей переменной в полученных моделях.

В первой построенной модели (Код 2) показатель *p-value* $= 1.3 \cdot 10^{-6}$, а также регрессор получил оценку в «***». Это говорит о сильной значимости регрессора *Education* при прогнозе объясняемой переменной *Agriculture*. Так как коэффициент $k = -1.5105$, связь между регрессором и объясняемой переменной — отрицательная.

Аналогично, во второй построенной модели (Код 3) показатель *p-value* $= 9.95 \cdot 10^{-8}$, а также регрессор получил оценку в «***». Это говорит о сильной значимости регрессора *Examination* при описании переменной *Agriculture*. Так как коэффициент $k = -1.9544$, связь между регрессором и объясняемой переменной — отрицательная.

Вывод. На основе предложенного набора данных можно построить линейные регрессии, которые будут давать приблизительные прогнозы объясняемой переменной *Agriculture* по значениям регрессоров *Education* и *Examination*. Перечисленные переменные значимы при описании показателя *Agriculture* и отрицательно коррелируют с ним, что говорит об отрицательной взаимосвязи показателя доли населения, занимающегося сельским хозяйством, и показателей успеваемости и экзаменационных оценок сельского населения Швейцарии на момент 1888 года.

Задача 2

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: Swiss.

Объясняемая переменная: *Education*.

Регрессоры: *Fertility*, *Agriculture*, *Infant.Mortality*.

Задача 2.1

1. Проверить, что в наборе данных нет линейной зависимости (построить зависимости между переменными, указанными в варианте, и проверить, что R^2 в каждой из них невысокий). В случае, если R^2 большой, один из таких столбцов можно исключить из рассмотрения.

Чтобы проверить, присутствует ли линейная зависимость между регрессорами, постоим парные модели, что представлено в коде 4.

```
lm(Fertility~Agriculture, data) # R^2 = 0.12
lm(Fertility~Infant.Mortality, data) # R^2 = 0.17
lm(Agriculture~Infant.Mortality, data) # R^2 = 0.003
```

Код 4. Построение парных зависимостей между регрессорами.

Как видно из коэффициентов корреляции R^2 (Код 4), все используемые регрессоры имеют незначительную линейную зависимость друг от друга, поэтому стоит использовать каждый из них при построении модели.

2. Построить линейную модель зависимой переменной от указанных в варианте регрессоров по методу наименьших квадратов (команда *lm* пакета *lmtest* в языке R). Оценить, насколько хороша модель, согласно 1) R^2, 2) p-значениям каждого коэффициента.

```
> summary(lm(Education ~ Fertility + Agriculture + Infant.Mortality, data))

...

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      43.34806    6.81384   6.362 1.09e-07 ***
Fertility        -0.42527    0.08564  -4.966 1.13e-05 ***
Agriculture      -0.18549    0.04290  -4.323 8.95e-05 ***
Infant.Mortality  0.34385    0.34428   0.999    0.324
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.995 on 43 degrees of freedom
Multiple R-squared:  0.6366,    Adjusted R-squared:  0.6112
F-statistic: 25.11 on 3 and 43 DF,  p-value: 1.522e-09
```

Код 5. Построение зависимости
Education $\sim$ *Fertility* $+$ *Agriculture* $+$ *Infant.Mortality*.

Полученная «сырая» модель (Код 5) имеет $R^2 = 0.64$, что говорит о том, что зависимость между регрессорами и объясняемой переменной определенно прослеживается, но она не очень сильна. В то же время регрессоры *Fertility* и *Agriculture* имеют оценку в «***», что говорит о сильной статистической связанности этих регрессоров и объясняемой переменной. Регрессор *Infant.Mortality* же имеет оценку в ноль «*», поэтому им можно пренебречь.

```
> summary(lm(Education ~ Fertility + Agriculture, data))

...

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 47.92166    5.04541   9.498 3.19e-12 ***
Fertility   -0.38515    0.07563  -5.092 7.10e-06 ***
Agriculture -0.19596    0.04160  -4.711 2.49e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.995 on 44 degrees of freedom
Multiple R-squared:  0.6281,    Adjusted R-squared:  0.6112
F-statistic: 37.16 on 2 and 44 DF,  p-value: 3.534e-10
```

Код 6. Построение зависимости $Education \sim Fertility + Agriculture$.

В сравнении с предыдущей моделью (Код 5) мы потеряли 1% в R^2, поэтому отбросим регрессор *Infant.Mortality* (Код 6).

3. Ввести в модель логарифмы регрессоров (если возможно). Сравнить модели и выбрать наилучшую.

В коде 7 представлено построение модели со всевозможными комбинациями логарифмов регрессоров.

```
lm(Education ~ Fertility + Agriculture, data) # R^2 = 0.6281
lm(Education ~ log(Fertility) + Agriculture, data) # R^2 = 0.6875
lm(Education ~ Fertility + log(Agriculture), data) # R^2 = 0.6994
lm(Education ~ log(Fertility) + log(Agriculture), data) # R^2 = 0.7301
```

Код 7. Введение логарифмов регрессоров в модель.

Как видно из построенных моделей (Код 7), наилучшей оказалась та, в которой логарифмированы оба регрессора: *Fertility* и *Agriculture*.

4. Ввести в модель всевозможные произведения пар регрессоров, в том числе квадраты регрессоров. Найти одну или несколько наилучших моделей по доле объяснённого разброса в данных R^2.

В коде 8 представлено построение моделей с введенными в них произведениями и квадратами регрессоров.

```
lm(Education ~ log(Fertility) + log(Agriculture), data) # R^2 = 0.7301
lm(Education ~ I(log(Fertility)*log(Agriculture)), data) # R^2 = 0.6405
lm(Education ~ I(log(Fertility)^2) + log(Agriculture), data) # R^2 = 0.7227
lm(Education ~ log(Fertility) + I(log(Agriculture)^2), data) # R^2 = 0.7098
lm(Education ~ I(log(Fertility)^2) + I(log(Agriculture)^2),
  data) # R^2 = 0.6983
```

Код 8. Введение произведений и квадратов регрессоров в модель.

Анализируя полученные результаты (Код 8), видим, что изначальная модель с двумя логарифмами регрессоров остается лучшей при оценке по R^2, однако также достаточно «хорошей» вышла модель $Education \sim I(\log(Fertility)^2) + \log(Agriculture)$.

5. Выбрать одну из лучших моделей. Перечислить все регрессоры x в ней и построить парные регрессии $y = a + bx$. Если переменная x значима, сделать вывод о наличии взаимосвязи между объясняемой переменной y и рассматриваемой объясняющей переменной x: отрицательной (если коэффициент $b < 0$) или положительной (если коэффициент $b > 0$).

Выберем модель $Education \sim \log(Fertility) + \log(Agriculture)$. Построим парные регрессии между объясняемой переменной и регрессором для каждого регрессора выбранной модели (Код 9, Код 10).

```
> summary(lm(Education ~ log(Fertility), data))

...

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     162.819     21.118   7.710 9.14e-10 ***
log(Fertility)  -35.870      4.984  -7.198 5.20e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.628 on 45 degrees of freedom
Multiple R-squared:  0.5351,    Adjusted R-squared:  0.5248
F-statistic:  51.8 on 1 and 45 DF,  p-value: 5.195e-09
```

Код 9. Построение парной регрессии $Education \sim \log(Fertility)$.

Согласно построенной модели (Код 9), регрессор $\log(Fertility)$ сильно значим при описании поведения переменной *Education* ($p = 5.2 \cdot 10^{-9}$), взаимосвязь между ним и объясняемой переменной — строго отрицательная ($k = -35.87$).

```
> summary(lm(Education ~ log(Agriculture), data))

...

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)        46.461      4.686   9.915 6.78e-13 ***
log(Agriculture)   -9.472      1.226  -7.726 8.66e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.374 on 45 degrees of freedom
Multiple R-squared:  0.5702,    Adjusted R-squared:  0.5606
F-statistic: 59.69 on 1 and 45 DF,  p-value: 8.664e-10
```

Код 10. Построение парной регрессии $Education \sim \log(Agriculture)$.

Аналогично выводу, сделанному по предыдущей модели (Код 9), регрессор $\log(Agriculture)$ сильно значим при описании поведения переменной *Education* ($p = 8.66 \cdot 10^{-10}$), взаимосвязь между ним и

объясняемой переменной — отрицательная ($k = -9.472$) (Код 10).

Вывод. Объясняемая переменная *Education* может быть описана поведением регрессоров *Fertility* и *Agriculture* в силу их значимости, в то время как регрессор *Infant.Mortality* - не значим при описании переменной *Education*. При этом наибольшая корреляция прослеживается при описании переменной *Education* через логарифмы соответствующих переменных: $\log(\textit{Fertility})$ и $\log(\textit{Agriculture})$. Связь между упомянутыми регрессорами и объясняемой переменной - строго отрицательная, что говорит об отрицательной зависимости показателя успеваемости сельского населения и показателей рождаемости и доли населения занимающегося сельским хозяйством в провинциальных районах Швейцарии на момент 1888 года.

Задача 2.2

1. Для лучшей зависимости, построенной при решении практического задания 2.1, оценить доверительные интервалы для всех коэффициентов в модели, $p = 95\%$.

Для начала найдем значение t-критерия Стьюдента для построения доверительных интервалов коэффициентов регрессоров полученной в первой части задачи лучшей модели (Код 11).

```
# t-критерий для p = 95%, 42 степеней свободы
t_crit = qt(0.975, df=44) # t_crit = 2.015
```

Код 11. Нахождение t-критерия Стьюдента для лучшей модели полученной в задаче 2.1.

Далее построим доверительные интервалы вида $(k - \textit{t-crit} \cdot \textit{Std. err.}; k + \textit{t-crit} \cdot \textit{Std. err.})$, что представлено в коде 12.

```
# log(Fertility)
c(-22.874 - t_crit*4.479, -22.874 + t_crit*4.479) # (-31.90083, -13.84717)
# log(Agriculture)
c(-6.461 - t_crit*1.146, -6.461 + t_crit*1.146) # (-8.770611, -4.151389)
```

Код 12. Нахождение доверительных интервалов коэффициентов регрессоров.

2. Сделать вывод об отвержении или невозможности отвергнуть статистическую гипотезу о том, что коэффициент равен 0.

Анализируя полученные интервалы (Код 12), можно заключить, что ни в один из построенных интервалов не попадает 0, что свидетельствует об опровержении гипотезы о равенстве коэффициента нулю.

3. Доверительный интервал для одного прогноза ($p = 95\%$, набор значений регрессоров выбрать самим).

Построим прогноз для следующего набора регрессоров: $\textit{Fertility} = 80$, $\textit{Agriculture} = 25$ (Код 13).

```
> best_model = lm(Education ~ log(Fertility) + log(Agriculture), data)
> new.data = data.frame(Fertility = 80, Agriculture = 25)
> predict(best_model, new.data, interval="confidence")
       fit      lwr      upr
1 10.97556 8.287011 13.66411
```

Код 13. Построение прогноза для лучшей модели из задачи 2.1.

Согласно построенному прогнозу (Код 13), с вероятностью 95% значение объясняемой переменной *Education* будет лежать в интервале $(8.287; 13.664)$, наиболее вероятное значение объясняемой переменной при выбранных значениях регрессоров — 10.976.

4. Для каждого регрессора x (или функции его первой степени), участвующего в модели, построить парную регрессию $y = a + bx$,

здесь y – объясняемая переменная. Для каждой парной зависимости оценить доверительный интервал коэффициента b. Указать выявленную связь между объясняемой переменной и регрессором: положительная (доверительный интервал не содержит 0, наиболее вероятное значение коэффициента положительное) / отрицательная (доверительный интервал не содержит 0, наиболее вероятное значение коэффициента отрицательное) / отсутствует (не опровергается гипотеза о том, что коэффициент равен 0).

Построим парные регрессии между индивидуальными регрессорами и объясняемой переменной, найдем t-критерии для каждой из них и построим довереительные интервалы коэффициентов при индивидуальных регрессорах (Код 14, Код 15).

```r
# 45 DF, k = -35.87, Std. err. = 4.984
summary(lm(Education~log(Fertility), data))
# t_criterion = 2.014 (p = 95%, 45 степеней свободы)
t_crit = qt(0.975, df=45)
# (-45.90829, -25.83171)
c(-35.87 - t_crit*4.984, -35.87 + t_crit*4.984)
```

Код 14. Нахождение доверительного интервала для коэффициента при регрессоре $\log(Fertility)$ в модели $Education \sim \log(Fertility)$.

Полученный доверительный интервал (Код 14) — полностью отрицательный (не содержит 0), а следовательно выявленная связь между объясняемой переменной *Education* и регрессором $\log(Fertility)$ — отрицательная.

```r
# 45 DF, k = -9.472, Std. err. = 1.226
summary(lm(Education~log(Agriculture), data))
# t_criterion = 2.014 (p = 95%, 45 степеней свободы)
t_crit = qt(0.975, df=45)
(-11.941291, -7.002709)
c(-9.472 - t_crit*1.226, -9.472 + t_crit*1.226)
```

Код 15. Нахождение доверительного интервала для коэффициента при регрессоре $\log(Agriculture)$ в модели $Education \sim \log(Agriculture)$.

Аналогично предыдущему результату, полученный доверительный интервал (Код 15) — полностью отрицательный (не содержит 0), а следовательно выявленная связь между объясняемой переменной *Education* и регрессором $\log(Agriculture)$ — отрицательная.

Вывод. Продолжая вывод задачи 2.1, была опровергнута гипотеза о равенстве нулю коэффициентов при регрессорах $\log(Fertility)$ и $\log(Agriculture)$, что подтвердило значимость и строго отрицательную взаимосвязь упомянутых регрессоров и объясняемой переменной *Education*. Аналогичным образом была опровергнута гипотеза о равенстве коэффициента нулю в парных регрессиях между объясняемой переменной и упомянутыми регрессорами.

Задача 3

Необходимо проанализировать данные волны мониторинга экономического положения и здоровья населения РФ (данные обследования РМЭЗ НИУ ВШЭ).

Набор данных: 20 волна выборки РМЭЗ НИУ ВШЭ.

Подмножества респондентов: не женатые мужчины, с высшим образованием; городские жители, состоящие в браке.

Перед началом выполнения поставленных задач прочитаем и обработаем необходимые для построения моделей данные из датасета (Приложение 3).

1. Построить линейную регрессию зарплаты на все параметры, выделенные из данных мониторинга. Оценить коэффициент вздутия дисперсии VIF.

Так как при разбиении переменной *p-marst* на бинарные переменные *wed1*, *wed2* и *wed3* одна из них становится на прямую линейно зависимой от двух других, исключим самую малоинформативную из них — *wed3*. Осуществив данный шаг, можно приступить к построению модели и оценке коэффициента вздутия дисперсии, что представлено в коде 16.

```
> model1 = lm(salary~sex+wed1+wed2+city_status+higher_educ+
    salary_satisfaction+is_entrepreneur+if_subordinates+age+
    week_duration+last_vacation, data=clean_data)
> summary(model1) # Adjusted R-squared:  0.2751
> vif(model1)
sex     wed1    wed2    city_status    higher_educ    salary_satisfaction
1.14    1.56    1.68    1.02           1.12           1.03
is_entrepreneur    if_subordinates    age     week_duration    last_vacation
1.03               1.09               1.18    1.07             1.02
```

Код 16. Вычисление коэффициента вздутия дисперсии VIF для модели построенной от всех регрессоров.

Все полученные при вызове команды *vif* коэффициенты, достаточно

близки к единице, что говорит об отсутствии мультиколлинеарности между выбранными регрессорами (Код 16).

2. Поэкспериментировать с функциями вещественных параметров: использовать логарифмы, степени, произведения вещественных регрессоров.

Для упрощения задачи перебора оснований логарифма и показательной функции напишем функции, автоматически осуществляющие подбор оптимальных параметров (Код 17, Код 18).

```
best_log_base <- function(var, regressor) {
  best_base <- 0.1
  best_r_squared <- 0.0
  for (base in seq(from=0.1, to=5.0, by=0.1)) {
    if (base != 1.0) {
      predictor <- log(regressor, base)
      model = lm(var ~ predictor)
      r_squared <- summary(model)$r.squared
      if (r_squared > best_r_squared) {
        best_r_squared <- r_squared
        best_base <- base
      }
    }
  }
  best_base
}
```

Код 17. Функция для вычисления оптимального основания логарифма регрессора в парной регрессии.

```r
best_exp_base <- function(var, regressor) {
  best_base <- 0.1
  best_r_squared <- 0.0
  for (base in seq(from=0.1, to=3.0, by=0.1)) {
    predictor <- base^regressor
    model = lm(var ~ predictor)
    r_squared <- summary(model)$r.squared
    if (r_squared > best_r_squared) {
      best_r_squared <- r_squared
      best_base <- base
    }
  }
  best_base
}
```

Код 18. Функция для вычисления оптимального основания показательной функции от регрессора в парной регрессии.

Теперь используем созданные функции для подбора оптимальных оснований логарифма и показательной функции (Код 17, Код 18) для переменных *age* и *week_duration*, что представлено в коде 19.

```r
> best_log_base(clean_data$salary, clean_data$age)
2.4
> best_log_base(clean_data$salary, clean_data$week_duration)
3.5
> best_exp_base(clean_data$salary, clean_data$age)
1.6
> best_exp_base(clean_data$salary, clean_data$week_duration)
0.8
```

Код 19. Вычисление оснований логарифма и показательной функции для переменных *age* и *week_duration*.

Наилучшие значения оснований логарифма и показательной функции для переменных *age* и *week_duration* (Код 19) представлены в таблице 1.

Таблица 1. Оптимальные значения оснований логарифма и показательной функции для переменных *age* и *week_duration*.

	$\log_a x$	a^x
age	2.4	1.6
week_duration	3.5	0.8

Используем полученные значения оснований при построении улучшенной модели, а также исключим регрессоры *wed2* и *last_vacation*, так как они не значимы по p-метрике (Код 20).

```
model3 = lm(salary~sex+wed1+city_status+higher_educ+salary_satisfaction+
  is_entrepreneur+if_subordinates+I(2.3^age)+I(0.7^week_duration),
  data=clean_data)
summary(model3) # Adjusted R-squared:  0.2798
```

Код 20. Построение улучшенной модели с использованием логарифмов и показательных функций от регрессоров.

Из кода 20 видно, что R^2 незначительно возрос — приблизительно на 0.5% по сравнению с первой построенной моделью (Код 16). Также регрессор *wed1* полностью потерял значимость, поэтому исключим его в последующих моделях.

Теперь введем квадраты и произведения регрессоров в модель (Код 21).

```
summary(lm(salary~sex+city_status+higher_educ+salary_satisfaction+
  is_entrepreneur+if_subordinates+I(2.3^age*0.7^week_duration),
  data=clean_data)) # Adjusted R-squared:  0.2718
summary(lm(salary~sex+city_status+higher_educ+salary_satisfaction+
  is_entrepreneur+if_subordinates+I((2.3^age)^2)+I(0.7^week_duration),
  data=clean_data)) # Adjusted R-squared:  0.2734
summary(lm(salary~sex+city_status+higher_educ+salary_satisfaction+
  is_entrepreneur+if_subordinates+I(2.3^age)+I((0.7^week_duration)^2),
  data=clean_data)) # Adjusted R-squared:  0.277
```

Код 21. Построение моделей с введенными в них квадратами и произведениями вещественных регрессоров.

Согласно полученным в результате выполнения кода 21 значениям $R^2_{\text{adj.}}$,

ни одна из построенных моделей не оказалась лучше оригинальной при сравнении по коэффициенту корреляции R^2.

3. Выделить наилучшие модели из построенных: по значимости параметров, включённых в зависимости, и по объяснённому с помощью построенных зависимостей разбросу adjusted R^2 - R^2_{adj}.

Наилучшей по значимости включенных параметров и по объясненному разбросу $R^2_{\text{adj.}}$ оказалась вторая построенная модель — с введенными в нее логарифмами и показательными функциями регрессоров (Код 20). Также неплохими вышли модели с произведениями и квадратами регрессоров (Код 21).

4. Для каждого регрессора x (в первой степени, не его функции), участвующего в лучшей модели, построить парную регрессию $y = a + bx$. Указать значимость переменной x, построить доверительный интервал для её коэффициента, указать наличие положительной / отрицательной взаимосвязи между ней и объясняемой переменной. Сделать вывод о том, какие индивиды получают наибольшую зарплату.

Для упрощения задачи построения доверительного интервала для регрессора в каждой из данных моделей напишем функцию, автоматизирующую данный процесс (Код 22).

```
interval <- function(model){
  df <- df.residual(model)
  t_crit <- qt(0.975, df)
  out = summary(model)
  std_err <- out$coefficients[2, 2]
  koef <- out$coefficients[2, 1]
  c(koef - std_err*t_crit, koef + std_err*t_crit)
}
```

Код 22. Функция построения доверительного интервала при коэффициенте регрессора в парной модели.

Теперь, пользуясь функцией из кода 22, построим доверительные интервалы для коэффициентов в парных регрессиях всех чистых переменных, что представлено в коде 23.

```
interval(lm(salary~sex, data=clean_data)) # (0.446; 0.567)
interval(lm(salary~city_status, data=clean_data)) # (0.334; 0.469)
interval(lm(salary~higher_educ, data=clean_data)) # (0.481; 0.605)
interval(lm(salary~salary_satisfaction, data=clean_data)) # (0.528; 0.653)
interval(lm(salary~is_entrepreneur, data=clean_data)) # (0.443; 0.781)
interval(lm(salary~if_subordinates, data=clean_data)) # (0.566; 0.706)
interval(lm(salary~age, data=clean_data)) # (-0.173; -0.113)
interval(lm(salary~week_duration, data=clean_data)) # (0.110; 0.171)
```

Код 23. Построенные доверительные интервалы коэффициентов в парных регрессиях.

Согласно выводу кода 23, ни в один из построенных доверительных интервалов не попал 0, что опровергает гипотезу о нахождении нуля в доверительном интервале и говорит о том, что целевая переменная *salary* имеет связь с каждой из объясняющих ее переменных. Также каждый из полученных доверительных интервалов за исключением интервала коэффициента при переменной *age* оказался полностью положительным, что приводит нас к следующему выводу: наибольшую зарплату получают молодые мужчины, проживающие в городе, с высшим образованием, удовлетворенные своей заработной платой, предприниматели, имеющие подчиненных, а также имеющие длинную рабочую неделю.

5. Проверить лучшую модель на указанных подмножествах респондентов. Построить доверительные интервалы для оставшихся в модели коэффициентов и указать, попадает ли 0 в них.

Для отбора соответствующих подмножеств респондентов воспользуемся кодом 24.

```r
# Подмножество не женатых мужчин с высшим образованием
subset1 = subset(clean_data, sex == 1)
subset1 = subset(subset1, wed1 == 0)
subset1 = subset(subset1, higher_educ == 1)

# Подмножество городских жителей, состоящих в браке
subset2 = subset(clean_data, city_status == 1)
subset2 = subset(subset2, wed1 == 1)
```

Код 24. Отбор подмножеств респондентов из датасета.

Теперь, используя полученные в коде 24 подмножества респондентов, построим линейную регрессию с лучшим отобранным набором регрессоров для каждого из них (Код 25).

```r
summary(lm(salary~city_status+salary_satisfaction+is_entrepreneur+
  if_subordinates+I(2.3^age)+I(0.7^week_duration),
  data=subset1)) # Adjusted R-squared:  0.07123

summary(lm(salary~sex+higher_educ+salary_satisfaction+is_entrepreneur+
  if_subordinates+I(2.3^age)+I(0.7^week_duration),
  data=subset2)) # Adjusted R-squared:  0.2708
```

Код 25. Построение моделей на отобранных подмножествах респондентов.

Анализируя полученные показатели $R^2_{\text{adj.}}$ построенных в коде 25 моделей, можно заключить, что отобранная в пункте 3 задачи модель дает крайне неточные прогнозы на первом отобранном подмножестве респондентов (женатые мужчины с высшим образованием) что выражается в падении показателя $R^2_{\text{adj.}}$ на $\approx 21\%$ (с 27.98% до 7.123%), в то время как на втором отобранном подмножестве респондентов (городские жители состоящие в браке) модель дает достаточно точные прогнозы с падением показателя $R^2_{\text{adj.}}$ на $\approx 1\%$ (с 27.98% до 27.08 %).

В заключение, проверим доверительные интервалы оставшихся регрессоров — функции переменных *age* и *week_duration* (Код 26).

```r
interval(lm(salary~I(2.3^age), data=clean_data)) # (-0.026; -0.017)
interval(lm(salary~I(0.7^week_duration), data=clean_data)) # (-1.883; -1.252)
```

Код 26. Построение доверительных интервалов для оставшихся участвующих в модели регрессоров.

Ни в один из построенных в коде 26 интервалов не попал 0, что говорит о присутствии связи между регрессорами 2.3^{age} и $0.7^{week\_duration}$ и объясняемой переменной *salary*.

Вывод. Из предоставленного датасета были успешно отобраны и обработаны необходимые для предсказания заработной платы с приемлемой точностью данные. Наилучшая из построенных моделей была оценена по метрике $R^2_{\text{adj.}}$ в 0.2798. Доверительные интервалы коэффициентов при регрессорах, содержащихся в лучшей модели, при построении парных регрессий между ними и объясняемой переменной не содержали в себе 0, что гарантировало их связь с объясняемой переменной. По результату исследования было выявлено, что наибольшую заработную плату получают молодые мужчины, проживающие в городе, с высшим образованием, удовлетворенные своей заработной платой, предприниматели, имеющие подчиненных, а также имеющие длинную рабочую неделю. Полученная модель дает неточные прогнозы на подмножестве женатых мужчин с высшим образованием и точные прогнозы на подмножестве городских жителей, состоящих в браке.

Задача 4

Необходимо загрузить данные из указанного набора и произвести следующие действия.

Набор данных: Video game sales

Тип классификатора: DecisionTreeClassifier (решающее дерево)

Классификация по столбцу (целевой признак): Platform (DS – класс 0, остальные уровни – класс 1)

1. Обработать указанный набор данных, подготовив его к решению задачи классификации. Выделить целевой признак и удалить его из данных, на основе которых будет обучаться классификатор. Разделить набор данных на тестовую и обучающую выборку. Построить классификатор указанного типа для задачи классификации по указанному параметру. Оценить точность построенного классификатора с помощью метрик *precision*, *recall* и *F1* на тестовой выборке.

Чтобы приступить к построению решающего дерева, прочитаем датасет из файла, отбросим отсутствующие значения (NA), отделим целевой признак в отдельный массив и удалим его из оригинального датасета, а также разделим датасет на тренировочную и тестовую выборки при помощи функции *train_test_split* пакета *sklearn* с параметром *test_size* = 0.25 (Код 27).

```
data = pd.read_csv('./vgsales.csv')
data['Platform'] = np.where(data['Platform'] == 'DS', 0, 1)
data = data.dropna()

platform = data.loc[:, data.columns.isin(['Platform'])]
features = ['Rank', 'Year', 'NA_Sales', 'EU_Sales',
            'JP_Sales', 'Other_Sales', 'Global_Sales']
x = data.loc[:, data.columns.isin(features)]

x_tr, x_val, y_tr, y_val = train_test_split(x, platform,
                                            test_size=0.25, random_state=8)
```

Код 27. Чтение, обработка и разделение датасета на тренировочную и тестовую выборки.

Теперь построим решающие деревья различной глубины (от 1 до 20) и выберем наилучшее из них, ориентируясь на большие значения, получаемые при вызове функции *score* (Код 28).

```
best_tree = DecisionTreeClassifier(random_state=6679, max_depth=1)
best_tree = best_tree.fit(x_tr, y_tr)

for i in range(2,10):
    tree = DecisionTreeClassifier(random_state=6679, max_depth=i)
    tree = tree.fit(x_tr, y_tr)
    if tree.score(x_val, y_val) > best_tree.score(x_val, y_val):
        best_tree = tree
```

Код 28. Построение оптимального решающего дерева.

Далее выведем метрики *precision*, *recall* и *F1* полученного классификатора при помощи функции *classification_report* (Код 29).

```
> report = classification_report(y_val, best_tree.predict(x_val))
> print(report)
              precision    recall  f1-score   support

           0       1.00      0.01      0.01       545
           1       0.87      1.00      0.93      3528

    accuracy                           0.87      4073
   macro avg       0.93      0.50      0.47      4073
weighted avg       0.88      0.87      0.81      4073
> accuracy = accuracy_score(y_val, best_tree.predict(x_val))
> print(f'Accuracy: {accuracy}')
Accuracy: 0.8671740731647435
```

Код 29. Вывод отчета по полученному классификатору.

Анализируя полученные значения метрик (Код 29), можно заключить, что полученный классификатор достаточно точен на данных, представленных в датасете (*accuracy* ≈ 0.87), однако он не опознает класс 0, что выражается в значениях метрик *recall* и *F1* равных 0.01. Значение метрики *accuracy* равное ≈ 0.87 говорит о том, что построенный классификатор будет давать верные прогнозы приблизительно в 87% случаев.

Также оценим вклад каждой из переменных в описании целевого признака использовав код 30.

```
feature_importances = (best_tree.feature_importances_ /
                       sum(best_tree.feature_importances_))
results = pd.DataFrame({'Features': features,
                        'Importances': feature_importances})
results.sort_values(by='Importances', inplace=True)
ax = plt.barh(results['Features'], results['Importances'])
plt.xlabel('Feature importances')
plt.show()
```

Код 30. Построение графика важности признаков при описании целевого признака.

При вызове кода 30 получим график, представленный на рисунке 1.

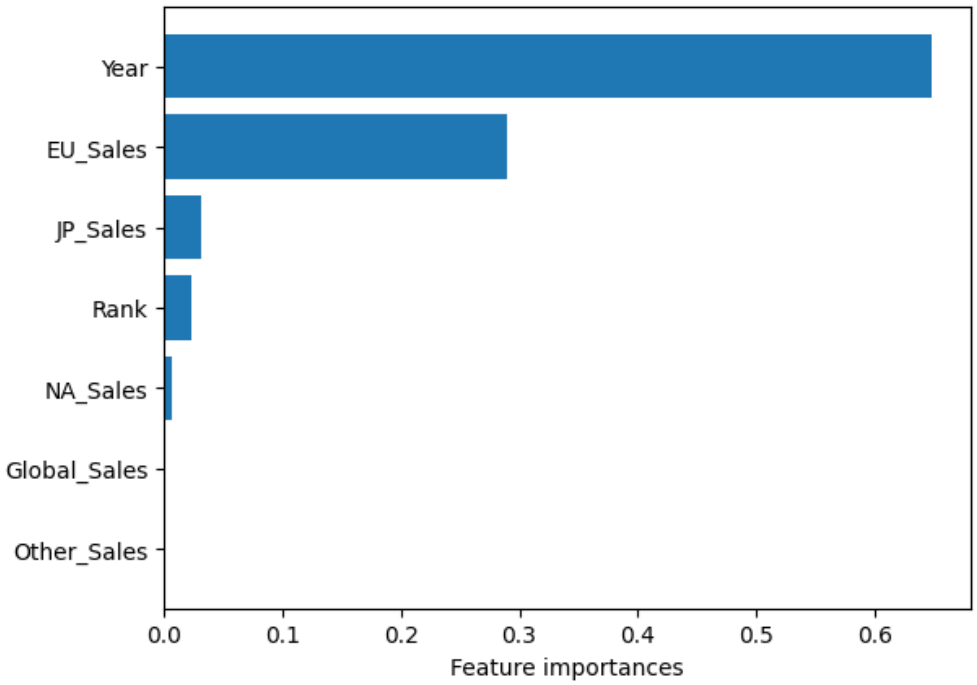

Рис. 1. График важности признаков при описании целевого признака для классификатора DecisionTreeClassifier.

Согласно изображенному на рисунке 1 графику, наибольший вклад в предсказание целевого признака вносят признаки *Year* и *EU_Sales*.

Схему усеченного полученного решающего дерева можно увидеть в Приложении 6.

Согласно структуре построенного решающего дерева (усеченная схема изображена в приложении 6), можно построить следующую систему неравенств, ограничивающих объекты класса 0 и 1 (1):

$$\textit{Class 0}: \left[\begin{cases} \textit{Year} \leq 1988.5 \\ \textit{Rank} > 15836.5 \end{cases} \begin{cases} \textit{Year} > 2003.5 \\ \left[\begin{cases} \textit{EU\_Sales} \leq 0.0 \\ \textit{NA\_Sales} > 0.02 \end{cases} \right. \\ \textit{EU\_Sales} \leq 3.06 \end{cases} \right. \tag{1}$$

В данном случае достаточно одной системы неравенств, так как классы 0 и 1 — взаимоисключающие, что значит, что все объекты класса 0 будут удовлетворять системе, а объекты класса 1 — нет.

2. Построить классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Оценить его качество с помощью метрик *precision*, *recall* и *F1* на тестовой выборке. С помощью *GridSearch* перебрать различные комбинации гиперпараметров. Определить, какой из классификаторов оказывается лучше.

Для начала построим Случайный Лес, не подбирая гиперпараметры, и выведем уже использованные в пункте 1 задачи метрики (Код 31).

```
> forest = RandomForestClassifier(random_state=6679)
> forest = forest.fit(x_tr, np.ravel(y_tr))
> report = classification_report(y_val, forest.predict(x_val))
> print(report)
> accuracy = accuracy_score(y_val, forest.predict(x_val))
> print(f'Accuracy: {accuracy}')
              precision    recall  f1-score   support

           0       0.46      0.33      0.39       545
           1       0.90      0.94      0.92      3528

    accuracy                           0.86      4073
   macro avg       0.68      0.64      0.65      4073
weighted avg       0.84      0.86      0.85      4073
Accuracy: 0.859317456420329
```

Код 31. Построение классификатора RandomForestClassifier и вывод его метрик.

Построенный классификатор получил большую на 4% оценку по метрике *F1* в общем случае, а также точность прогнозов в подмножестве класса 0 значительно увеличилась (метрика *F1* увеличилась на 0.40).

График важности признаков при описании целевого признака изображен на рисунке 2.

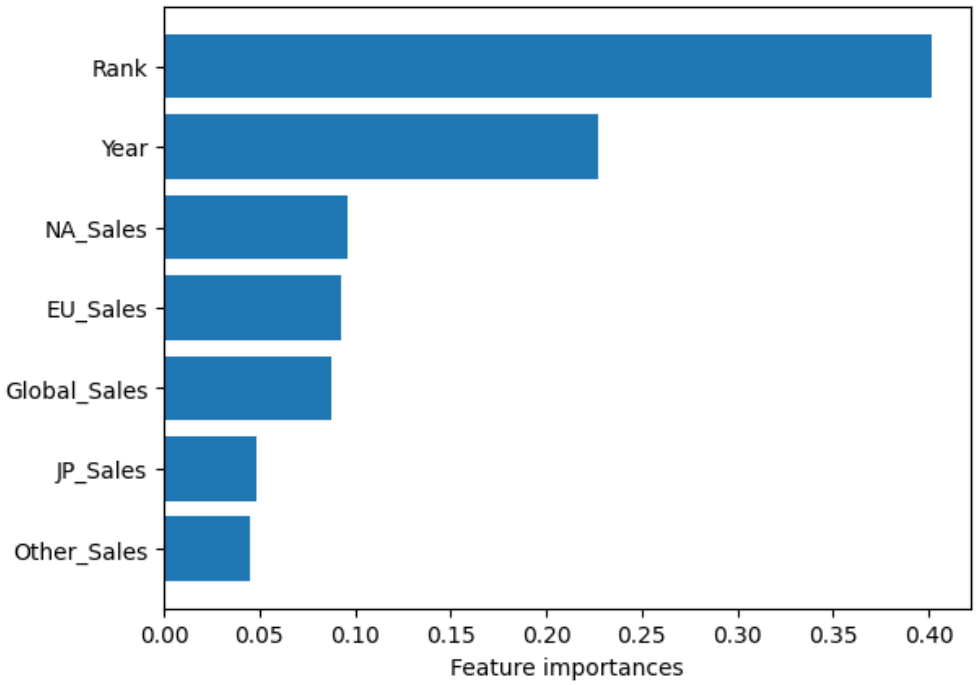

Рис. 2. График важности признаков при описании целевого признака для классификатора RandomForestClassifier.

Согласно рисунку 2, главными признаками при описании целевого признака стали *Rank* и *Year*.

Теперь, используя метод *GridSearch*, подберем следующие гиперпараметры классификатора: *num_estimators* и *max_depth* (Код 32).

```
param_grid = {
    'n_estimators': [50, 100, 200, 400],
    'max_depth': list(range(1, 20)),
    'criterion': ['gini']
}
tuned_forest = GridSearchCV(estimator=RandomForestClassifier(),
                            param_grid=param_grid, cv=5, refit=True)
tuned_forest.fit(x_tr, np.ravel(y_tr))
print(f'n_estimators: {tuned_forest.best_estimator_.n_estimators}')
print(f'max_depth: {tuned_forest.best_estimator_.max_depth}')
```

Код 32. Подбор гиперпараметров для классификатора RandomForestClassifier.

При многократном вызове кода 32 оказалось, что оптимального количества решающих деревьев в случайном лесу (*num_estimators*) не существует, так как результатом работы программы в каждом случае было

новое число, поэтому было принято зафиксировать значение в 100 решающих деревьев. Оптимальная глубина решающего дерева в построенном классификаторе оказалась равной 9.

Построим классификатор типа Случайный Лес с подобранными гиперпараметрами и оценим его согласно использованным ранее метрикам (Код 33).

```
> report = classification_report(y_val,
                                 tuned_forest.best_estimator_.predict(x_val))
> print(report)
              precision    recall  f1-score   support

           0       0.67      0.08      0.15       545
           1       0.88      0.99      0.93      3528

    accuracy                           0.87      4073
   macro avg       0.77      0.54      0.54      4073
weighted avg       0.85      0.87      0.83      4073
> accuracy = accuracy_score(y_val, tuned_forest.best_estimator_.predict(x_val))
> print(f'Accuracy: {accuracy}')
Accuracy: 0.8718389393567395
```

Код 33. Построение классификатора RandomForestClassifier с подобранными гиперпараметрами и вывод его метрик.

Построенный в коде 33 классификатор получил худшую оценку по метрике *F1*, чем изначальный классификатор, построенный без подбора гиперпараметров (потеря в 1%), но получил лучшую оценку по критерию *accuracy* (выигрыш в 1%). Также в случае классификатора с введенными гиперпараметрами резко упала точность в подмножестве класса 0, что прослеживается в падении метрики *F1* в соответствующем классе до показателя в 15%.

Также приведем график важности признаков при описании целевого признака для полученного классификатора (Рис. 3).

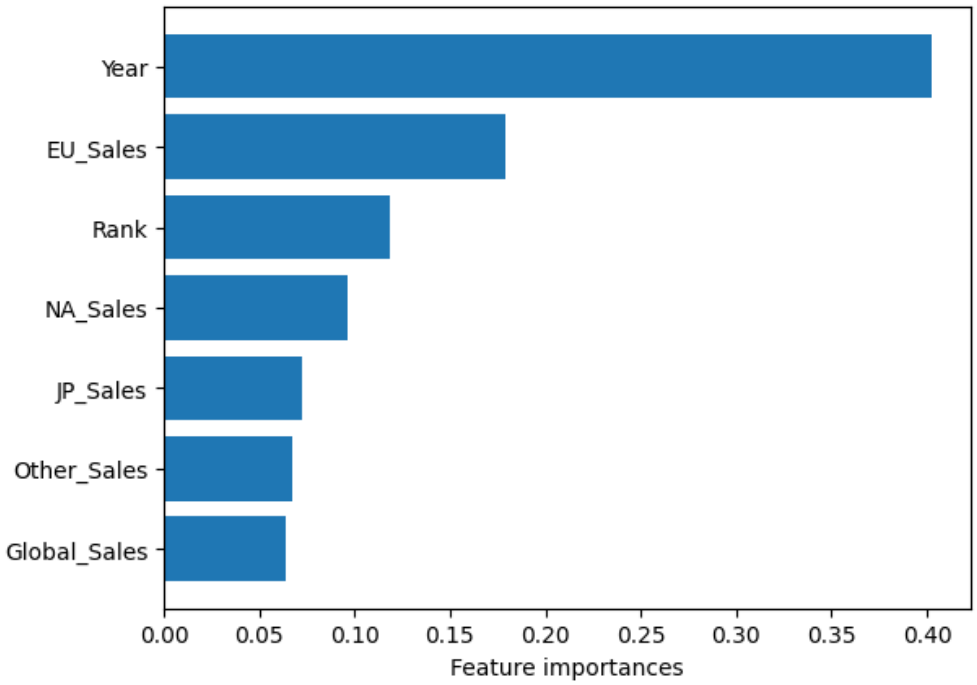

Рис. 3. График важности признаков при описании целевого признака классификатора RandomForestClassifier с введенными в него гиперпараметрами.

Согласно рисунку 3, главными признаками при описании целевого признака усовершенствованным классификатором стали *Year* и *EU_Sales*.

Вывод. Данные, необходимые для выполнения задачи классификации, были успешно прочитаны и обработаны. Первый построенный классификатор (решающее дерево) оказался довольно точным на данных, представленных в датасете, однако его оценка на подмножестве класса 0 оказалась крайне низкой ($F1 = 0.01$), что говорит о том, что этот класс вообще не опознается построенным классификатором. Точность в общем случае же составила 47%. Второй построенный классификатор (Случайный Лес) оказался более точным в общем случае ($F1_{macro\ avg} = 65\%$), а также класс 0 начал расспознаваться классификатором ($F1_0 = 40\%$). После подбора гиперпараметров для классификатора типа Случайный Лес показатели точности классификатора при оценке по метрике *F1* снова упали ($F1_{macro\ avg} = 54\%$), класс 0 вновь перестал распознаваться. Во время построения указанных классификаторов наиболее важными признаками

при описании целевого признака *Platform* оказались *Year*, *Rank* и *EU_Sales*.

Задача 5

Необходимо провести анализ нижеуказанного датасета и сделать обработку данных по предложенному алгоритму.

Набор данных: Denver Crime Data

1. Сколько в наборе данных объектов и признаков? Дать описание каждому признаку, если оно есть.

Если отбросить всяческие ID, в наборе присутствует 16 признаков со следующими описаниями (нумерация идет по столбцам; столбцы, содержащие описания и ID — отброшены):

3) *offense_code* — код преступления;

4) *offense_code_extension* — расширение кода преступления (стоит «склеить» с *offence_code* или отбросить);

5) *first_occurrence_date* — когда происшествие произошло в первый раз;

6) *last_occurrence_date* — когда происшествие произошло в последний раз относительно описываемого происшествия;

7) *reported_date* — когда о происшествии было доложено в полицию;

8) *incident_address* — адрес улицы, на которой произошло происшествие;

9) *geo_x* — гео-код по оси *X*;

10) *geo_y* — гео-код по оси *Y*;

11) *geo_lat* — широта, по которой произошло происшествие;

12) *geo_lon* — долгота, по которой произошло происшествие;

13) *district_id* — округ;

14) *precinct_id* — участок;

15) *neighborhood_id* — район;

16) *is_crime* — 1, если преступление; 0, если нет;

17) *is_traffic* — 1, если дорожно транспортное происшествие; 0, если нет;

18) *victim_count* — число пострадавших в происшествии.

2. Сколько категориальных признаков представлено в датасете, какие?

Из перечисленных в пункте 1 признаков категориальными являются: *offense_code*, *offense_code_extension*, *incident_address*, *district_id*, *precinct_id*, *neighborhood_id*.

3. Указать столбец с максимальным количеством уникальных значений категориального признака.

Для того, чтобы подсчитать количество уникальных значений в каждом из столбцов содержащих категориальные признаки, прочитаем датасет из файла, отберем из него все перечисленные в пункте 1 признаки (для дальнейшей обработки), и выделим отдельный *pandas.DataFrame* с категориальными признаками. Наконец, выведем суммарное количество объектов в датасете и количества уникальных значений по столбцам (Код 34).

```
data = pd.read_csv('./crime.csv', encoding='unicode_escape')
features = ['offense_code', 'offense_code_extension', 'first_occurrence_date',
            'last_occurrence_date', 'reported_date', 'incident_address',
            'geo_x', 'geo_y', 'geo_lon', 'geo_lat', 'district_id',
            'precinct_id', 'neighborhood_id', 'is_crime', 'is_traffic',
            'victim_count']
data = data.loc[:, data.columns.isin(features)]
criterial_features = ['offense_code', 'offense_code_extension',
                      'incident_address', 'district_id', 'precinct_id',
                      'neighborhood_id']
criterial_data = data.loc[:, data.columns.isin(criterial_features)]
print(f'Number of rows: {len(data)}')
criterial_data.nunique().sort_values(ascending=False)
```

Код 34. Подсчет количества уникальных значений категориальных признаков по столбцам в датасете.

Выполнение кода 34 дает результат, представленный в коде 35.

```
Number of rows: 386865
incident_address          90518
offense_code                141
neighborhood_id              78
precinct_id                  41
district_id                   8
offense_code_extension        7
```

Код 35. Количество уникальных значений соответствующих категориальных признаков в датасете.

Как видно из кода 35, максимальное количество значений (90518) достигается у категориального признака *incident_addresses* (адрес, по которому произошло преступление), что не удивительно. Проводить классификацию по этому признаку, учитывая, что суммарное количество объектов в датасете равно 386865 — бессмысленно, если предварительно не проводить обработку данной переменной.

4. Есть ли бинарные признаки?

В датасете присутствуют бинарные признаки. Из перечисленных в пункте

1 признаков бинарными являются: *is_crime*, *is_traffic*.

5. Какие числовые признаки представлены в датасете?

Из перечисленных в пункте 1 признаков числовыми являются: *first_occurence_date*, *last_occurrence_date*, *reported_date*, *geo_x*, *geo_y*, *geo_lat*, *geo_lon*, *victim_count*.

6. Есть ли пропуски?

В датасете присутствуют пропуски. Количество пропусков в каждом из столбцов приведено ниже.

7. Сколько объектов с пропусками присутствует в датасете?

В коде 36 представлено нахождение количества объектов с пропусками, присутствующих в датасете.

```
# Total number of objects: 386865
print(f'Total number of objects: {len(data)}')
# Number of objects with NA values: 180345
print(f'Number of objects with NA values: {len(data) - len(data.dropna())}')
# Number of "clean" objects: 206520
print(f'Number of \"clean\" objects: {len(data.dropna())}')
```

Код 36. Подсчет количества объектов с пропусками в датасете.

Согласно выводу кода 36, количество объектов с пропусками оказалось равным 180345 (46.6% от суммарного количества объектов).

8. Указать столбец с максимальным количеством пропусков.

Для нахождения количества пропусков в каждом из столбцов воспользуемся алгоритмом, представленным в коде 37.

```
> print('Number of NA values per column:')
> for col in sorted(data.columns,
                    key=lambda x: data[x].isna().sum(),
                    reverse=True):
>     print(f'{col}: {data[col].isna().sum()}')
Number of NA values per column:
last_occurrence_date: 175556
geo_lon: 15769
geo_lat: 15769
incident_address: 15503
geo_x: 15503
geo_y: 15503
neighborhood_id: 689
district_id: 57
offense_code: 0
...
```

Код 37. Подсчет количества пропусков в каждом из столбцов.

Согласно выводу кода 37, столбцом с максимальным количеством пропусков оказался столбец *last_occurence_date*. Также много пропусков присутствует в столбцах, относящихся к местоположению происшествия.

9. Выявить возможные выбросы, аномальные значения.

Чтобы проверить данные на выбросы, отделим время происшествия от даты его происхождения в отдельный признак, а также выделим числовые признаки в отдельный *pandas.DataFrame* (Приложение 7).

Теперь получим количество выбросов, присутствующих в датасете (Код 38).

```
for feature in normalizible_features:
    Q1 = normalizible_data[feature].quantile(0.25)
    Q3 = normalizible_data[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    normalizible_data_no_outliers = normalizible_data[
        (normalizible_data[feature] ⩾
         lower_bound) & (normalizible_data[feature] ⩽ upper_bound)]
normalizible_data_no_outliers = normalizible_data_no_outliers.dropna()
print(f"Number of outliers: {len(normalizible_data) -
                             len(normalizible_data_no_outliers)}")
print(f'Total number of objects: {len(normalizible_data)}')
Number of outliers: 15769
Total number of objects: 386865
```

Код 38. Подсчет количества выбросов, содержащихся в датасете.

После отбора выбросов по межквартильному диапозону (Код 38) оказалось, что приблизительно каждый 25-ый объект — выброс. При построении моделей и классификаторов исключим указанные объекты из датасета.

10. Указать столбец с максимальным средним значением после нормировки признаков через стандартное отклонение.

Применим нормализацию числовых признаков через среднее квадратическое отклонение (Код 39).

```
> std_scaler = preprocessing.StandardScaler()
> normalized_numerical_data = pd.DataFrame(
>     std_scaler.fit_transform(normalizible_data.dropna()),
>     columns=normalizible_data.dropna().columns,
>     index=normalizible_data.dropna().index)
> print("Mean values:")
> for col in normalizible_features:
>     print(f'{col}: {normalized_numerical_data[col].mean()}')
Mean values:
geo_x: 5.650704189145687e-16
geo_y: -9.493152402339494e-16
geo_lat: 5.0441993575674176e-15
geo_lon: 1.0637079181535481e-14
victim_count: -3.342133424421861e-17
reported_time_utc: 8.612383926124051e-17
```

Код 39. Подсчет средних значений по столбцам после применения нормализации через среднее квадратическое отклонение.

Согласно коду 39, столбцом с максимальным средним значением после нормировки признаков через стандартное квадратическое отклонение оказался столбец *geo_lon*, однако все полученные средние значения очень близки к нулю.

11. Указать столбец с целевым признаком.

Столбцом с целевым признаком является столбец *offence_code* (после обработки данных — столбец *offence_code+extension*) — тип происшествия.

12. Сколько объектов попадает в тренировочную выборку при использовании train_test_split с параметрами *test_size* = 0.3, *random_state* = 42?

Возьмем столбец *offence_code* как столбец с целевым признаком, и разделим датасет на тренировочную и целевую выборки с указанными параметрами (Код 40).

```
> binary_features = ['is_crime', 'is_traffic']
> features = criterial_features + numerical_features + binary_features
> x_features = features.copy()
> x_features.remove('offense_code')
> x_features.remove('offense_code_extension')
> target = data['offense_code']
> x = data.loc[:, data.columns.isin(x_features)]
> x_tr, x_val, target_tr, target_val = train_test_split(
>     x, target, test_size=0.3, random_state=42)
> print(f'Training split size: {len(x_tr)}')
> print(f'Validation split size: {len(x_val)}')
Training split size: 270805
Validation split size: 116060
```

Код 40. Разделение датасета на тренировочную и тестовую выборки.

Согласно выводу кода 40, в тренировочную выборку попало 270805 объектов, в тестовую — 116060.

13. Между какими признаками наблюдается линейная зависимость (корреляция)?

В коде 41 представлен код для нахождения коррелирующих признаков.

```
for col in normalizible_features:
    print(normalized_numerical_data.loc[:].corr()[col][:], "\n")
```

Код 41. Нахождение коррелирующих признаков, представленных в датасете.

Согласно выводу кода 41 (Приложение 8), сильная линейная зависимость прослеживается между следующими признаками:

- *first_occurence_date* / *last_occurence_date* и *reported_date*;
- *geo_lat* и *geo_y*;
- *geo_lon* и *geo_x*;
- *reported_time* и *reported_time_utc*;

При дальнейшей обработке исключим некоторые из перечисленных признаков.

14. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

Чтобы применить метод главных компонент, исключим мультиколлинеарные признаки, закодируем признак *district_id*, применив one-hot-encoding, и объединим и закодируем признаки *offense_code* и *offense_code_extension*, применив label-encoding (Приложение 9).

```
pca_data = clean_data.drop(columns=['offence_code+extension'])
pca = PCA()
x_pca = pca.fit(pca_data.values)
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance_ratio)
n_components = np.argmax(cumulative_variance ⩾ 0.9) + 1
plt.barh(pca.get_feature_names_out(), explained_variance_ratio)
plt.xlabel('Feature importances')
plt.show()
plt.bar(range(1, len(cumulative_variance) + 1), cumulative_variance)
plt.xlabel('Cummulative variance by number of features')
plt.show()
# 4 features are sufficient to explain 90% of the variance.
print(f'{n_components} features are sufficient to explain 90% of the variance.')
```

Код 42. Применение метода PCA на обработанных данных.

Согласно выводу кода 42, 4-ех признаков оказалось достаточно для объяснения 90% дисперсии после применения метода PCA. Также были сгенерированы 2 графика: график доли дисперсии объясненной каждой из компонент (Приложение 10) и график суммарной доли объясненной дисперсии в зависимости от числа взятых компонент, изображенный на рисунке 4.

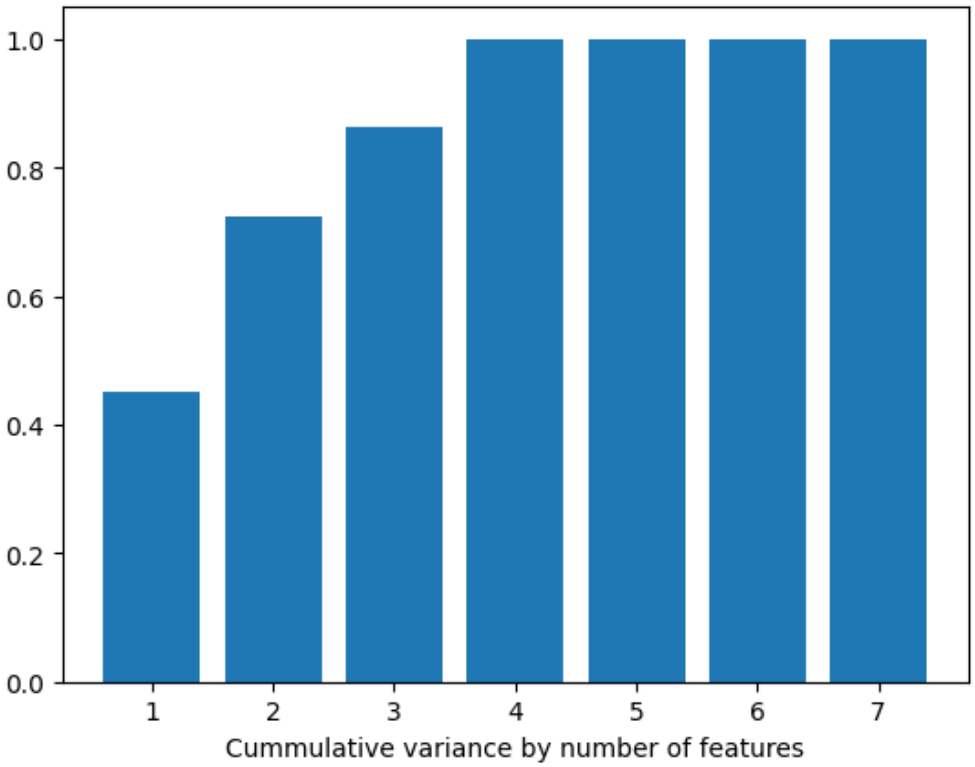

Рис. 4. График суммарной доли объясненной дисперсии в зависимости от числа взятых компонент после применения метода PCA.

15. Какой признак вносит наибольший вклад в первую компоненту?

Для того, чтобы выяснить, какой признак вносит наибольший вклад в первую компоненту, воспользуемся кодом, представленным в коде 43.

```
> loadings = pca.components_[0]
> feature_importances = pd.Series(
>     loadings, index=pca_data.columns)
> print('Feature contributions to the first component (pca0):')
> print(feature_importances.abs().sort_values(ascending=False))
Feature contributions to the first component (pca0):
district_id          0.998685
geo_y                0.037739
geo_x                0.029419
reported_time_utc    0.018063
victim_count         0.003628
is_crime             0.000000
is_traffic           0.000000
```

Код 43. Доли вкладов в первую компоненту признаками после применения метода PCA.

Согласно выводу кода 43, наибольший вклад в 99% в первую компоненту вносит признак *district_id*.

16. Построить двухмерное представление данных с помощью алгоритма t-SNE. Оценить, на сколько кластеров визуально разделяется выборка. Объяснить смысл кластеров.

В коде 44 представлено построение двумерного представления данных при помощи алгоритма t-SNE.

```
shuffled_data = shuffle(clean_data, random_state=0).head(20000)
x = shuffled_data.drop(columns=['offence_code+extension',])
y = shuffled_data.loc[:, clean_data.columns.isin(['offence_code+extension'])]
z = TSNE(n_components=2, learning_rate='auto', random_state=0,
         init='pca', perplexity=50).fit_transform(x)
```

Код 44. Применение алгоритма t-SNE для построения двумерного представления данных, содержащихся в датасете.

Теперь построим графики с разной окраской точек, соответствующей одному из признаков, для определения смысла кластеров (Код 45).

```
sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['district_id'],
                data=df).set(title='t-SNE Visualization (district_id)')
plt.show()
sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['victim_count'],
                data=df).set(title='t-SNE Visualization (victim_count)')
plt.show()
sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['offence_code+extension'],
                data=df).set(
                    title='t-SNE Visualization (offence_code+extension)')
plt.show()
```

Код 45. Построение трех окрашенных графиков для визуализации кластеров, полученных после применения алгоритма t-SNE.

В результате выполнения кода 45 получим 3 графика (Рис. 5, Приложение 11, Приложение 12).

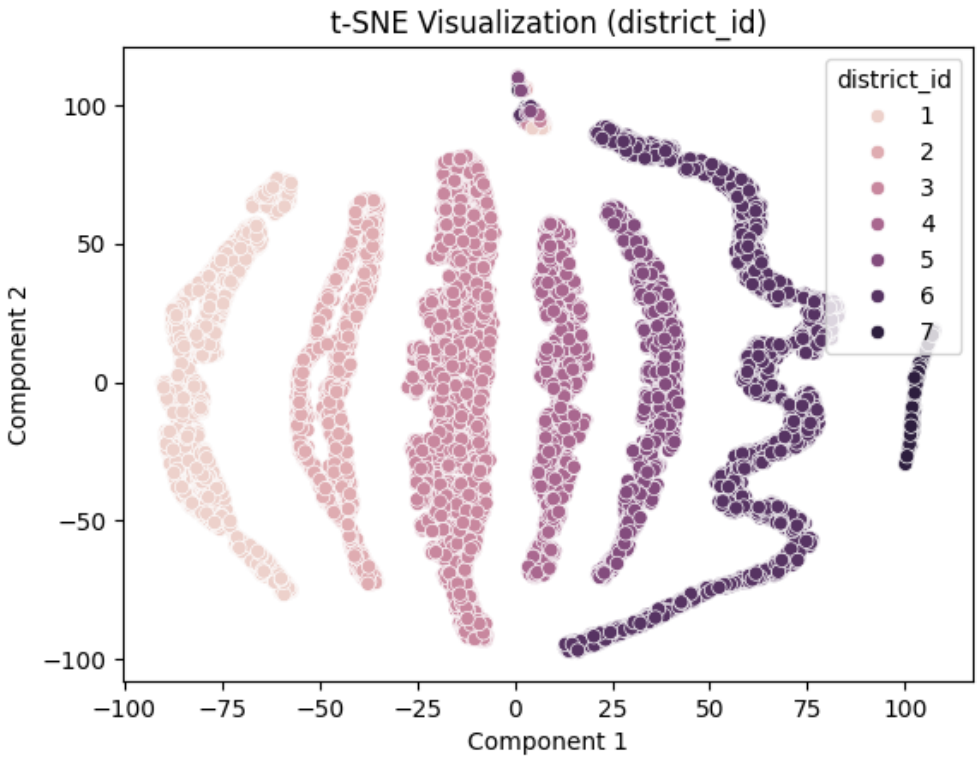

Рис. 5. Двумерная визуализация данных, полученная после применения алгоритма t-SNE (окраска по критерию *district_id*).

Анализируя первый полученный график, изображенный на рисунке 5, можно заключить, что 7 из 8-ми отчетливо выраженных на графике кластеров представляют собой районы города (с 1-го по 7-ой). Последний изображенный на графиках кластер представляет собой преступления с большим количеством жертв, что можно отчетливо увидеть на втором построенном графике (Приложение 11). Большинство из преступлений попавших в 8-ой кластер — преступления с кодом вида 131**, который соответствует преступлениям — различным видам массовой стрельбы.

Вывод. Предоставленный для выполнения задачи 6 датасет был успешно прочитан из файла, из датасета были отброшены индексные переменные и переменные, содержащие описания. В датасете были выделены категориальные, бинарные и числовые признаки. Были выявлены категориальные признаки с максимальным количеством уникальных значений а также столбцы с максимальным количеством пропусков. По итогам анализа было определено, что около трети объектов,

присутствующих в датасете имеют пропуски, а также что приблизительно каждый 80-ый объект является выбросом. После нормализации числовых признаков через среднее квадратическое отклонение у каждого из представленных признаков среднее значение оказалось крайне близко к нулю. Столбцом с целевым признаком был определен столбец *offence_code*. Было обнаружено значительное количество коррелирующих признаков, которые при дальнейшей обработке были отброшены. После применения метода PCA на полученных обработанных данных оказалось, что 4-ех компонент достаточно для объяснения 90% дисперсии в данных. Наибольший вклад в первую компоненту после применения метода PCA имел признак *district_id*. После применения алгоритма t-SNE данные визуально разбились на 8 кластеров, 7 из которых представляли собой районы города (*district_id*), а последний 8-ой кластер представлял собой преступления с большим количеством жертв.

Заключение

В задаче 1 была выявлена отрицательная связь между объясняемой переменной *Agriculture* и регрессорами *Education* (k = −1.5105) и *Examination* (k = −1.9544), коэффициенты корреляции R^2 получились равными 0.409 и 0.4713 соответственно, что говорит о неточности даваемых моделями прогнозов. Используемые коэффициенты также были оценены как значимые по p-метрике (*p-value* $= 1.3 \cdot 10^{-6}$ в случае регрессора *Education*, *p-value* $= 9.95 \cdot 10^{-8}$ в случае регрессора *Examination*).

В задаче 2 лучшей из построенных моделей оказалась модель $\textit{Education} \sim \log(\textit{Fertility}) + \log(\textit{Agriculture})$ с показателем $R^2 = 0.7301$. Также при построении доверительных интервалов коэффициентов регрессоров была опровергнута гипотеза о равенстве коэффициента нулю, что гарантировало значимость указанных регрессоров и подтвердило отрицательную взаимосвязь объясняемой переменной *Education* и регрессоров $\log(\textit{Fertility})$ и $\log(\textit{Agriculture})$ ($k_{\log(\textit{Fer.})} = -35.87$, ($k_{\log(\textit{Agr.})} = -9.472$).

В задаче 3 наилучшая из построенных моделей была оценена по метрике $R^2_{\text{adj.}}$ в 0.2798. Доверительные интервалы коэффициентов при регрессорах, содержащихся в лучшей модели, при построении парных регрессий между ними и объясняемой переменной не содержали в себе 0, что гарантировало их связь с объясняемой переменной. По результату исследования было выявлено, что наибольшую заработную плату получают молодые мужчины, проживающие в городе, с высшим образованием, удовлетворенные своей заработной платой, предприниматели, имеющие подчиненных, а также имеющие длинную рабочую неделю. Лучшая полученная модель дает неточные прогнозы на подмножестве женатых мужчин с высшим образованием и точные прогнозы на подмножестве городских жителей, состоящих в браке.

В задаче 4 первый построенный классификатор (решающее дерево) оказался довольно точным на данных, представленных в датасете, однако его оценка на подмножестве класса 0 оказалась крайне низкой ($F1 = 0.01$), что говорит о том, что этот класс вообще не опознается построенным классификатором. Точность в общем случае же составила 47%. Второй построенный классификатор (Случайный Лес) оказался более точным в общем случае ($F1_{macro\ avg} = 65\%$), а также класс 0 начал расспознаваться классификатором ($F1_0 = 40\%$). После подбора гиперпараметров для классификатора типа Случайный Лес показатели точности классификатора при оценке по метрике *F1* снова упали ($F1_{macro\ avg} = 54\%$), класс 0 вновь перестал распознаваться. Во время построения указанных классификаторов наиболее важными признаками при описании целевого признака *Platform* оказались *Year*, *Rank* и *EU_Sales*.

В задаче 5 в датасете были выделены категориальные, бинарные и числовые признаки. Были выявлены категориальные признаки с максимальным количеством уникальных значений а также столбцы с максимальным количеством пропусков. По итогам анализа было определено, что около трети объектов, присутствующих в датасете имеют пропуски, а также что приблизительно каждый 80-ый объект является выбросом. После нормализации числовых признаков через среднее квадратическое отклонение у каждого из представленных признаков среднее значение оказалось крайне близко к нулю. Столбцом с целевым признаком был определен столбец *offence_code*. Было обнаружено значительное количество коррелирующих признаков, которые при дальнейшей обработке были отброшены. После применения метода PCA на полученных обработанных данных оказалось, что 4-ех компонент достаточно для объяснения 90% дисперсии в данных. Наибольший вклад в первую компоненту после применения метода PCA имел признак *district_id*. После применения алгоритма t-SNE данные визуально разбились на 8

кластеров, 7 из которых представляли собой районы города (*district_id*), а последний 8-ой кластер представлял собой преступления с большим количеством жертв.

Список литературы

1. *Dougherty C.* Introduction to Econometrics [Текст] / Dougherty C. — Illustrated. — Oxford University Press, 2007. — 464 с.
2. *Hayashi F.* Econometrics [Текст] / Hayashi F. — 41 William Street, Princeton, New Jersey 08540 : Princeton University Press, 2000. — 669 с.
3. *McKinney W.* Python for Data Analysis [Текст] / McKinney W. — First edition. — 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc., 2012. — 452 с.
4. *Stock J.H.* Introduction to Econometrics [Текст] / Stock J.H., Watson M.W. — 4th edition. — Pearson, 2019. — 755 с.
5. *Магнус Я.Р.* Эконометрика. Начальный курс [Текст] / Магнус Я.Р., Катышев П.К., Персецкий А.А. — 6-е изд., перераб. и доп. — М. : Дело, 2004. — 576 с.

Приложение

Приложение 1. Код решения задачи 1

```r
library("lmtest")

mean(swiss$Agriculture) # Ср. арифм. Agriculture = 50.65957
var(swiss$Agriculture) # Дисперсия Agriculture = 515.7994
sd(swiss$Agriculture) # СКО Agriculture = 22.71122

mean(swiss$Education) # Ср. арифм. Education = 10.97872
var(swiss$Education) # Дисперсия Education = 92.45606
sd(swiss$Education) # СКО Education = 9.615407

mean(swiss$Examination) # Ср. арифм. Examination = 16.48936
var(swiss$Examination) # Дисперсия Examination = 63.64662
sd(swiss$Examination) # СКО Examination = 7.977883

# k = -1.5105, R^2 = 0.409
model1 = lm(formula=Agriculture~Education, data=swiss)
summary(model1)

# k = -1.9544, R^2 = 0.4713
model2 = lm(formula=Agriculture~Examination, data=swiss)
summary(model2)
```

Приложение 2. Код решения задачи 2

```r
library("lmtest")
library("GGally")
library("car")

data = swiss

# Все проверенные регрессоры не имеют значимой взаимной линейной зависимости
summary(lm(Fertility~Agriculture, data)) # R^2 = 0.12
summary(lm(Fertility~Infant.Mortality, data)) # R^2 = 0.17
summary(lm(Agriculture~Infant.Mortality, data)) # R^2 = 0.003

# R^2 = 0.63, исключим регрессор Infant.Mortality в силу высокого k-коэф.
summary(lm(Education ~ Fertility + Agriculture + Infant.Mortality, data))

# Введем логарифмы регрессоров:
lm(Education ~ Fertility + Agriculture, data) # R^2 = 0.6281
lm(Education ~ log(Fertility) + Agriculture, data) # R^2 = 0.6875
lm(Education ~ Fertility + log(Agriculture), data) # R^2 = 0.6994
lm(Education ~ log(Fertility) + log(Agriculture), data) # R^2 = 0.7301

# Наилучшей вышла модель:
summary(lm(Education ~ log(Fertility) + log(Agriculture), data)) # R^2 =
↪  0.7301

# Введем квадраты и произведения регрессоров:
summary(lm(Education ~ log(Fertility) + log(Agriculture), data)) # R^2 =
↪  0.7301
summary(lm(Education ~ I(log(Fertility)*log(Agriculture)), data)) # R^2 =
↪  0.6405
summary(lm(Education ~ I(log(Fertility)^2) + log(Agriculture), data)) # R^2 =
↪  0.7227
summary(lm(Education ~ log(Fertility) + I(log(Agriculture)^2), data)) # R^2 =
↪  0.7098
summary(lm(Education ~ I(log(Fertility)^2) + I(log(Agriculture)^2), data)) #
↪  R^2 = 0.6983

# Наилучшей вышла изначальная модель с логарифмами:
summary(lm(Education ~ log(Fertility) + log(Agriculture), data)) # R^2 =
↪  0.7301
best_model = lm(Education ~ log(Fertility) + log(Agriculture), data)

# Рассмотрим индивидуальные лин. регрессии, построенные на основе каждого из
↪  регрессоров:
# Регрессор log(Fertility) - значим; взаимосвязь между объясняемой и
↪  объясняющей переменной - отрицательная (k = -35)
summary(lm(Education ~ log(Fertility), data))
```

```r
# Регрессор log(Agriculture) - значим; взаимосвязь между объясняемой и объясняющей переменной - отрицательная (k = -9.4)
summary(lm(Education ~ log(Agriculture), data))

# Построим доверительные интервалы для коэффициетов регрессоров полученной модели:
# Std. err:
#   log(Fertility) : 4.479
#   log(Agriculture) : 1.146
summary(best_model)
t_crit = qt(0.975, df=44) # t_crit = 2.015 (p = 95%, 42 степени свободы)
# Ни в один из интервалов не попадает "0" ⟹ гипотеза о равенстве
# коэффициента нулю опровергнута
# log(Fertility)
c(-22.874 - t_crit*4.479, -22.874 + t_crit*4.479) # (-31.90083; -13.84717)
# log(Agriculture)
c(-6.461 - t_crit*1.146, -6.461 + t_crit*1.146) # (-8.770611; -4.151389)

# Доверительный интервал для одного прогноза: Fertility = 80, Agriculture = 25
best_model = lm(Education ~ log(Fertility) + log(Agriculture), data)
new.data = data.frame(Fertility = 80, Agriculture = 25)
predict(best_model, new.data, interval="confidence")
#       fit      lwr      upr
# 10.97556 8.287011 13.66411

# Построим модели для индивидуальных регрессоров лучшей модели
# Education ~ log(Fertility)
summary(lm(Education~log(Fertility), data))
t_crit2 = qt(0.975, df=45)# t_crit2 = 2.014 (p = 95%, 45 степеней свободы)
# Полученный дов. интервал - полностью отрицательный ⟹ выявленная связь между
# объясняемой переменной и регрессором - отрицательная
c(-35.87 - t_crit2*4.984, -35.87 + t_crit2*4.984) # (-45.90829; -25.83171)

# Education ~ log(Agriculture)
summary(lm(Education~log(Agriculture), data))
t_crit2 = qt(0.975, df=45) # t_crit2 = 2.014 (p = 95%, 45 степеней свободы)
# Полученный дов. интервал - полностью отрицательный ⟹ выявленная связь между
# объясняемой переменной и регрессором - отрицательная
c(-9.472 - t_crit2*1.226, -9.472 + t_crit2*1.226) # (-11.941291; -7.002709)
```

Приложение 3. Код решения задачи 3

```r
library("purrr")
library("lmtest")
library("dplyr")
library("car")
library("haven")
library("naniar")

# --- Чистка датасета от мусора ---

data <- read_sav("./r20i_os26c.sav")
View(data)
data1 <-
  select(data,
         pj13.2,
         ph5,
         p_marst,
         p_diplom,
         p_age,
         status,
         pj6.2,
         pj1.1.3,
         pj29,
         pj6,
         pj21b)

# Избавимся от зарезервированных значений по типу ЗАТРУДНЯЮСЬ ОТВЕТИТЬ
data1 = data1 %>% replace_with_na_all(condition = ~.x > 99999990)
# Проигнорируем значения NA
data1 = na.omit(data1)

write_sav(data1, "./r20i_os26c(stripped_of_NA).sav")

# --- One-hot-encoding и нормализация ---

data1 = read_sav("./r20i_os26c(stripped_of_NA).sav")

# Обработаем и нормализуем столбцы

clean_data = select(data1)

sex = as.numeric(map(data1$ph5, function(x) if(x == 1) 1 else 0))
clean_data["sex"] = sex
wed1 = as.numeric(map(data1$p_marst, function(x) if(x == 2 || x == 6) 1 else
↪  0))
clean_data["wed1"] = wed1
wed2 = as.numeric(map(data1$p_marst, function(x) if(x == 4 || x == 5) 1 else
↪  0))
```

```r
clean_data["wed2"] = wed2
wed3 = as.numeric(map(data1$p_marst, function(x) if(x == 1 || x == 3) 1 else 0))
clean_data["wed3"] = wed3
city_status = as.numeric(map(data1$status, function(x) if(x == 1 || x == 2) 1 else 0))
clean_data["city_status"] = city_status
higher_educ = as.numeric(map(data1$p_diplom, function(x) if(x >= 6) 1 else 0))
clean_data["higher_educ"] = higher_educ
salary_satisfaction = as.numeric(map(data1$pj1.1.3, function(x) if(x < 3) 1 else 0))
clean_data["salary_satisfaction"] = salary_satisfaction
is_entrepreneur = as.numeric(map(data1$pj29, function(x) if(x == 1) 1 else 0))
clean_data["is_entrepreneur"] = is_entrepreneur
if_subordinates = as.numeric(map(data1$pj6, function(x) if(x == 1) 1 else 0))
clean_data["if_subordinates"] = if_subordinates

sal = as.numeric(data1$pj13.2)
sal = (sal - mean(sal)) / sd(sal)
clean_data["salary"] = sal
age = as.numeric(data1$p_age)
age = (age - mean(age)) / sd(age)
clean_data["age"] = age
week_duration = as.numeric(data1$pj6.2)
week_duration = (week_duration - mean(week_duration)) / sd(week_duration)
clean_data["week_duration"] = week_duration
last_vacation = as.numeric(data1$pj21b)
last_vacation = (last_vacation - mean(last_vacation)) / sd(last_vacation)
clean_data["last_vacation"] = last_vacation

write_sav(clean_data, "./r20i_os26c(clean).sav")

# --- Построение модели ---

clean_data = read_sav("./r20i_os26c(clean).sav")

# Денормализуем week_duration, age и last_vacation для логарифмирования

min_age = min(clean_data$age)
clean_data["age"] = as.numeric(map(clean_data$age, function(x)
  x - min_age + 1e-8))
min_week_duration = min(clean_data$week_duration)
clean_data["week_duration"] = as.numeric(map(clean_data$week_duration, function(x)
  x - min_week_duration + 1e-8))
min_last_vacation = min(clean_data$last_vacation)
clean_data["last_vacation"] = as.numeric(map(clean_data$last_vacation, function(x)
  x - min_last_vacation + 1e-8))

```

```r
# Регрессор wed3 напрямую линейно зависим с регрессорами wed1, wed2; исключим его из модели
model1 = lm(
  salary ~ sex + wed1 + wed2 + wed3 + city_status + higher_educ + salary_satisfaction +
    is_entrepreneur + if_subordinates + age + week_duration + last_vacation,
  data = clean_data
)
summary(model1)

# Регрессоры wed2, last_vacation - не значимы; исключим их из модели
model2 = lm(
  salary ~ sex + wed1 + wed2 + city_status + higher_educ + salary_satisfaction +
    is_entrepreneur + if_subordinates + age + week_duration + last_vacation,
  data = clean_data
)
summary(model2)
vif(model2)

model2_1 = lm(
  salary ~ sex + wed1 + city_status + higher_educ + salary_satisfaction +
    is_entrepreneur + if_subordinates + age + week_duration,
  data = clean_data
)
summary(model2_1)
vif(model2_1)

# Напишем функции для нахождения оптимального основания логарифма и степенной функции

best_log_base <- function(var, regressor) {
  best_base <- 0.1
  best_r_squared <- 0.0

  for (base in seq(from=0.1, to=5.0, by=0.1)) {
    if (base != 1.0) {
      predictor <- log(regressor, base)
      model = lm(var ~ predictor)
      r_squared <- summary(model)$r.squared
      if (r_squared > best_r_squared) {
        best_r_squared <- r_squared
        best_base <- base
      }
    }
  }
  print("R^2: ")
  print(best_r_squared)
  best_base
}
```

```r

best_exp_base <- function(var, regressor) {
  best_base <- 0.1
  best_r_squared <- 0.0

  for (base in seq(from=0.1, to=3.0, by=0.1)) {
    predictor <- base^regressor
    model = lm(var ~ predictor)
    r_squared <- summary(model)$r.squared
    if (r_squared > best_r_squared) {
      best_r_squared <- r_squared
      best_base <- base
    }
  }
  print("R^2: ")
  print(best_r_squared)
  best_base
}

# Найдем с их помощью лучшие основания для логарифма и степенной функции для регрессоров age и week duration
# log_base for age
best_log_base(clean_data$salary, clean_data$age)
# log_base for week_duration
best_log_base(clean_data$salary, clean_data$week_duration)
# exp_base for age
best_exp_base(clean_data$salary, clean_data$age)
# exp_base for week_duration
best_exp_base(clean_data$salary, clean_data$week_duration)

# Видим, что лучшие показатели по R^2 дадут регресоры - I(2.3^age), I(0.7^week_duration)

# Использовав полученные регрессоры в модели, увидим, что R^2(adj) - увеличился (R^2(adj) = 0.2798)
model3 = lm(
  salary ~ sex + wed1 + city_status + higher_educ + salary_satisfaction +
    is_entrepreneur + if_subordinates + I(2.3 ^ age) + I(0.7 ^ week_duration),
  data = clean_data
)
summary(model3)

# Рассмотрим произведения и квадраты регрессоров
# R^2(adj) = 0.2718
summary(
  lm(
    salary ~ sex + wed1 + city_status + higher_educ + salary_satisfaction +
      is_entrepreneur + if_subordinates + I(2.3 ^ age * 0.7 ^ week_duration),
    data = clean_data
  )
```

```r
)
# R^2(adj) = 0.2734
summary(
  lm(
    salary ~ sex + wed1 + city_status + higher_educ + salary_satisfaction +
      is_entrepreneur + if_subordinates + I((2.3 ^ age) ^ 2) + I(0.7 ^
      ↪ week_duration),
    data = clean_data
  )
)
# R^2(adj) = 0.277
summary(
  lm(
    salary ~ sex + wed1 + city_status + higher_educ + salary_satisfaction +
      is_entrepreneur + if_subordinates + I(2.3 ^ age) + I((0.7 ^
      ↪ week_duration)^2),
    data = clean_data
  )
)

# Лучшей моделью оказалась изначальная модель; регрессор wed1 в ней
↪ окончательно потерял значимость, поэтому исключим его
summary(model3)

# У итоговой модели R^2(adj) = 0.2797
model4 = lm(
  salary ~ sex + city_status + higher_educ + salary_satisfaction +
  ↪ is_entrepreneur +
    if_subordinates + I(2.3 ^ age) + I(0.7 ^ week_duration),
  data = clean_data
)
summary(model4)

# Лучшая построенная модель - model3

# Нахождение доверительного интервала
interval <- function(model){
  df <- df.residual(model)
  t_crit <- qt(0.975, df)
  out = summary(model)
  std_err <- out$coefficients[2, 2]
  koef <- out$coefficients[2, 1]
  c(koef - std_err*t_crit, koef + std_err*t_crit)
}

# Построим парные регрессии для полученной модели и найдем доверительные
↪ интервалы соотв. коэффициентов:

# salary~sex: Связь - положительная (0 не попал в интервал ⟹ взаимосвязь
↪ присутствует)
```

```r
# Model
# summary(lm(salary~sex, data=clean_data))
# Interval
interval(lm(salary~sex, data=clean_data))

# salary~city_status: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~city_status, data=clean_data))
# Interval
interval(lm(salary~city_status, data=clean_data))

# salary~higher_educ: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~higher_educ, data=clean_data))
# Interval
interval(lm(salary~higher_educ, data=clean_data))

# salary~salary_satisfaction: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~salary_satisfaction, data=clean_data))
# Interval
interval(lm(salary~salary_satisfaction, data=clean_data))

# salary~is_entrepreneur: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~is_entrepreneur, data=clean_data))
# Interval
interval(lm(salary~is_entrepreneur, data=clean_data))

# salary~if_subordinates: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~if_subordinates, data=clean_data))
# Interval
interval(lm(salary~if_subordinates, data=clean_data))

# salary~age: Связь - отрицательная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
# summary(lm(salary~age, data=clean_data))
# Interval
interval(lm(salary~age, data=clean_data))

# salary~week_duration: Связь - положительная (0 не попал в интервал ⇒ взаимосвязь присутствует)
# Model
```

```r
# summary(lm(salary~week_duration, data=clean_data))
# Interval
interval(lm(salary~week_duration, data=clean_data))

# --- Вывод ---
# Таким образом, наибольшую зарплату получают молодые мужчины, проживающие в городе, с высшим образованием,
# удовлетворенные своей зарплатой, предприниматели, имеющие подчиненных, а также имеющие длинную рабочую неделю

# Проверим наилучшую построенную модель на подмножестве не женатых мужчин с высшим образованием (1)

subset1 = subset(clean_data, sex == 1)
subset1 = subset(subset1, wed1 == 0)
subset1 = subset(subset1, higher_educ == 1)
# View(subset1)

# Полученная модель дает неточные прогнозы на подмножестве (1) (R^2(adj) = 0.07123)
summary(
  lm(
    salary ~ city_status + salary_satisfaction + is_entrepreneur + if_subordinates +
      I(2.3 ^ age) + I(0.7 ^ week_duration),
    data = subset1
  )
)

# Проверим наилучшую построенную модель на подмножестве городских жителей, состоящих в браке (2)

subset2 = subset(clean_data, city_status == 1)
subset2 = subset(subset2, wed1 == 1)
# View(subset2)

# Полученная модель дает достаточно точные (в контексте данной задачи) прогнозы на подмножестве (2) (R^2(adj) = 0.2708)
summary(
  lm(
    salary ~ sex + higher_educ + salary_satisfaction + is_entrepreneur + if_subordinates +
      I(2.3 ^ age) + I(0.7 ^ week_duration),
    data = subset2
  )
)

# Проверим оставшиеся в модели коэффициенты
interval(lm(salary~age, data=clean_data))
interval(lm(salary~week_duration, data=clean_data))
```

```
313 # Ни в один из доверительных интервалов не попал 0 ⇒ взаимосвязь между
314 # регрессорами и объясняемой переменной - присутствует
```

Приложение 4. Код решения задачи 4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV


data = pd.read_csv('./vgsales.csv')
data['Platform'] = np.where(data['Platform'] == 'DS', 0, 1)
data = data.dropna()

platform = data.loc[:, data.columns.isin(['Platform'])]
features = ['Rank', 'Year', 'NA_Sales', 'EU_Sales',
            'JP_Sales', 'Other_Sales', 'Global_Sales']
x = data.loc[:, data.columns.isin(features)]

x_tr, x_val, y_tr, y_val = train_test_split(x, platform,
                                            test_size=0.25, random_state=8)

best_tree = DecisionTreeClassifier(random_state=6679, max_depth=1)
best_tree = best_tree.fit(x_tr, y_tr)

for i in range(2,10):
    tree = DecisionTreeClassifier(random_state=6679, max_depth=i)
    tree = tree.fit(x_tr, y_tr)
    if tree.score(x_val, y_val) > best_tree.score(x_val, y_val):
        best_tree = tree


report = classification_report(y_val, best_tree.predict(x_val))
print(report)

accuracy = accuracy_score(y_val, best_tree.predict(x_val))
print(f'Accuracy: {accuracy}')

feature_importances = (best_tree.feature_importances_ /
                       sum(best_tree.feature_importances_))

results = pd.DataFrame({'Features': features,
                        'Importances': feature_importances})
results.sort_values(by='Importances', inplace=True)

ax = plt.barh(results['Features'], results['Importances'])
plt.xlabel('Feature importances')
```

```
plt.show()

from sklearn import tree


fig = plt.figure(figsize=(10,7))
_ = tree.plot_tree(best_tree,
                   feature_names=features,
                   impurity=False,
                   proportion=True,
                   max_depth=2,
                   fontsize=12,
                   filled=True)
plt.show()


forest = RandomForestClassifier(random_state=6679)
forest = forest.fit(x_tr, np.ravel(y_tr))


report = classification_report(y_val, forest.predict(x_val))
print(report)

accuracy = accuracy_score(y_val, forest.predict(x_val))
print(f'Accuracy: {accuracy}')

feature_importances = (forest.feature_importances_ /
                       sum(forest.feature_importances_))

results = pd.DataFrame({'Features': features,
                        'Importances': feature_importances})
results.sort_values(by='Importances', inplace=True)

ax = plt.barh(results['Features'], results['Importances'])
plt.xlabel('Feature importances')
plt.show()


param_grid = {
    'n_estimators': [50, 100, 200, 400],
    'max_depth': list(range(1, 10)),
    'criterion': ['gini']
}
tuned_forest = GridSearchCV(estimator=RandomForestClassifier(),
                            param_grid=param_grid, cv=5, refit=True)
tuned_forest.fit(x_tr, np.ravel(y_tr))

print('Итерация 1')
 # n_estimators: значения варьируются от 50 до 400, выберем стандартные 100
print(f'n_estimators: {tuned_forest.best_estimator_.n_estimators}')
```

```python
print(f'max_depth: {tuned_forest.best_estimator_.max_depth}') # max_depth: 9

# Теперь сдвинем диапазон max_depth

param_grid = {
    'n_estimators': [100],
    'max_depth': list(range(5, 15)),
    'criterion': ['gini']
}
tuned_forest = GridSearchCV(estimator=RandomForestClassifier(),
                            param_grid=param_grid, cv=5, refit=True)
tuned_forest.fit(x_tr, np.ravel(y_tr))

print('Итерация 2')
print(f'max_depth: {tuned_forest.best_estimator_.max_depth}') # max_depth: 10

# Проанализируем полученный классификатор

report = classification_report(y_val,
                               tuned_forest.best_estimator_.predict(x_val))
print(report)

accuracy = accuracy_score(y_val, tuned_forest.best_estimator_.predict(x_val))
print(f'Accuracy: {accuracy}')

feature_importances = (tuned_forest.best_estimator_.feature_importances_ /
                       sum(tuned_forest.best_estimator_.feature_importances_))

results = pd.DataFrame({'Features': features,
                        'Importances': feature_importances})
results.sort_values(by='Importances', inplace=True)

ax = plt.barh(results['Features'], results['Importances'])
plt.xlabel('Feature importances')
plt.show()
```

Приложение 5. Код решения задачи 5

```
import pandas as pd
from scipy import stats
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.utils import shuffle
import seaborn as sns


data = pd.read_csv('./crime.csv', encoding='unicode_escape')
features = ['offense_code', 'offense_code_extension', 'first_occurrence_date',
            'last_occurrence_date', 'reported_date', 'incident_address',
            'geo_x', 'geo_y', 'geo_lon', 'geo_lat', 'district_id',
            'precinct_id', 'neighborhood_id', 'is_crime', 'is_traffic',
            'victim_count']
data = data.loc[:, data.columns.isin(features)]
criterial_features = ['offense_code', 'offense_code_extension',
                      'incident_address', 'district_id', 'precinct_id',
                      'neighborhood_id']
criterial_data = data.loc[:, data.columns.isin(criterial_features)]
print(f'Number of rows: {len(data)}')
criterial_data.nunique().sort_values(ascending=False)

print(f'Total number of objects: {len(data)}')
print(f'Number of objects with NA values: {len(data) - len(data.dropna())}')
print(f'Number of \'clean\' objects: {len(data.dropna())}')

print('Number of NA values per column:')
for col in sorted(data.columns,
                  key=lambda x: data[x].isna().sum(),
                  reverse=True):
    print(f'{col}: {data[col].isna().sum()}')

numerical_features = ['first_occurrence_date', 'last_occurrence_date',
                      'reported_date', 'geo_x', 'geo_y', 'geo_lat', 'geo_lon',
                      'victim_count', 'reported_time', 'reported_time_utc']
numerical_data = pd.DataFrame(index=data.index, columns=numerical_features)
numerical_data.loc[:, numerical_features] = data.loc[:,
    data.columns.isin(numerical_features)]

time_features = ['first_occurrence_date',
                 'last_occurrence_date', 'reported_date']

for col in time_features:
```

```
    numerical_data.loc[:, col] = numerical_data[col].apply(
        lambda timestr: pd.to_datetime(timestr, format='%m/%d/%Y %I:%M:%S
        ↪  %p'))

numerical_data['reported_time'] = \
    numerical_data['reported_date'].apply(lambda x: pd.to_datetime(
        x.strftime('%H:%M:%S')))

numerical_data.loc[:, 'reported_time_utc'] = \
    numerical_data['reported_time'].dt.hour * 60 * 60 + \
    numerical_data['reported_time'].dt.minute * 60 + \
    numerical_data['reported_time'].dt.second
numerical_data = numerical_data.astype('float64', errors='ignore')

# Отбросим значения даты и времени в формате Timestamp для нормализации
normalizible_features = ['geo_x', 'geo_y', 'geo_lat',
                         'geo_lon', 'victim_count', 'reported_time_utc']
normalizible_data = numerical_data.loc[:, numerical_data.columns.isin(
    normalizible_features)]

outliers = []
for col in normalizible_features:
    z_scores = stats.zscore(normalizible_data[col])
    outliers.extend([i for i, z in enumerate(z_scores) if abs(z) > 3])
print(f"Number of outliers: {len(outliers)}")
print(f'Number of \'clean\' objects: {len(data)}')

std_scaler = preprocessing.StandardScaler()
normalized_numerical_data = pd.DataFrame(
    std_scaler.fit_transform(normalizible_data.dropna()),
    columns=normalizible_data.dropna().columns,
    index=normalizible_data.dropna().index)
print("Mean values:")
for col in normalizible_features:
    print(f'{col}: {normalized_numerical_data[col].mean()}')
normalized_numerical_data.head()

binary_features = ['is_crime', 'is_traffic']
features = criterial_features + numerical_features + binary_features
x_features = features.copy()
x_features.remove('offense_code')
x_features.remove('offense_code_extension')
target = data['offense_code']
x = data.loc[:, data.columns.isin(x_features)]

x_tr, x_val, target_tr, target_val = train_test_split(
    x, target, test_size=0.3, random_state=42)
print(f'Training split size: {len(x_tr)}')
print(f'Validation split size: {len(x_val)}')

```

```
for col in normalizible_features:
    print(normalized_numerical_data.loc[:].corr()[col][:], "\n")

clean_features = ['offence_code+extension', 'reported_time_utc',
                  'geo_x', 'geo_y', 'district_id', 'is_crime',
                  'is_traffic', 'victim_count']
clean_data = pd.DataFrame(columns=clean_features, index=data.index)

clean_data['offence_code+extension'] = data['offense_code'] * \
    10 + data['offense_code_extension']
clean_data['reported_time_utc'] = numerical_data['reported_time_utc']
clean_data['geo_x'] = numerical_data['geo_x']
clean_data['geo_y'] = numerical_data['geo_y']
clean_data['district_id'] = data['district_id']
clean_data['is_crime'] = data['is_crime']
clean_data['is_traffic'] = data['is_traffic']
clean_data['victim_count'] = data['victim_count']
clean_data = clean_data.dropna()
# print(f'Number of objects in cleaned dataset: {len(clean_data.index)}')

normalizible_clean_features = ['reported_time_utc', 'geo_x', 'geo_y',
                               'victim_count']
normalizible_clean_data = clean_data.loc[:, clean_data.columns.isin(
    normalizible_clean_features)]
std_scaler = preprocessing.StandardScaler()
normalized_clean_data = pd.DataFrame(
    std_scaler.fit_transform(normalizible_clean_data),
    columns=normalizible_clean_data.columns,
    index=normalizible_clean_data.index)

for col in normalizible_clean_features:
    clean_data[col] = normalized_clean_data.loc[:, col]
clean_data = clean_data.apply(
    lambda x: pd.to_numeric(x, errors='coerce')).dropna()

clean_data.to_csv('./clean_crime.csv')

pca_data = clean_data.drop(columns=['offence_code+extension'])

pca = PCA()
x_pca = pca.fit(pca_data.values)
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance_ratio)
n_components = np.argmax(cumulative_variance >= 0.9) + 1

plt.barh(pca.get_feature_names_out(), explained_variance_ratio)
plt.xlabel('Feature importances')
plt.show()

plt.bar(range(1, len(cumulative_variance) + 1), cumulative_variance)
```

```
# plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance)
plt.xlabel('Cummulative variance by number of features')
plt.show()

print(f'{n_components} features are sufficient to explain 90% of the
↪  variance.')

loadings = pca.components_[0]
feature_importances = pd.Series(
    loadings, index=pca_data.columns)
print('Feature contributions to the first component (pca0):')
print(feature_importances.abs().sort_values(ascending=False))

shuffled_data = shuffle(clean_data, random_state=0).head(20000)

x = shuffled_data.drop(columns=['offence_code+extension',])
y = shuffled_data.loc[:, clean_data.columns.isin(['offence_code+extension'])]

z = TSNE(n_components=2, learning_rate='auto', random_state=0,
         init='pca', perplexity=50).fit_transform(x)

x_features = clean_features.copy()
x_features.remove('offence_code+extension')

df = pd.DataFrame()
df['y'] = y
df['Component 1'] = z[:, 0]
df['Component 2'] = z[:, 1]

sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['district_id'],
                data=df).set(title='t-SNE Visualization (district_id)')
plt.show()
sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['victim_count'],
                data=df).set(title='t-SNE Visualization (victim_count)')
plt.show()
sns.scatterplot(x='Component 1', y='Component 2',
                hue=shuffled_data['offence_code+extension'],
                data=df).set(
                    title='t-SNE Visualization (offence_code+extension)')
plt.show()

clean_data.sort_values(by=['victim_count'],
                       ascending=False).loc[:,
                        clean_data.columns.isin(['offence_code+extension',
                                                 'victim_count'])].head(10)
```

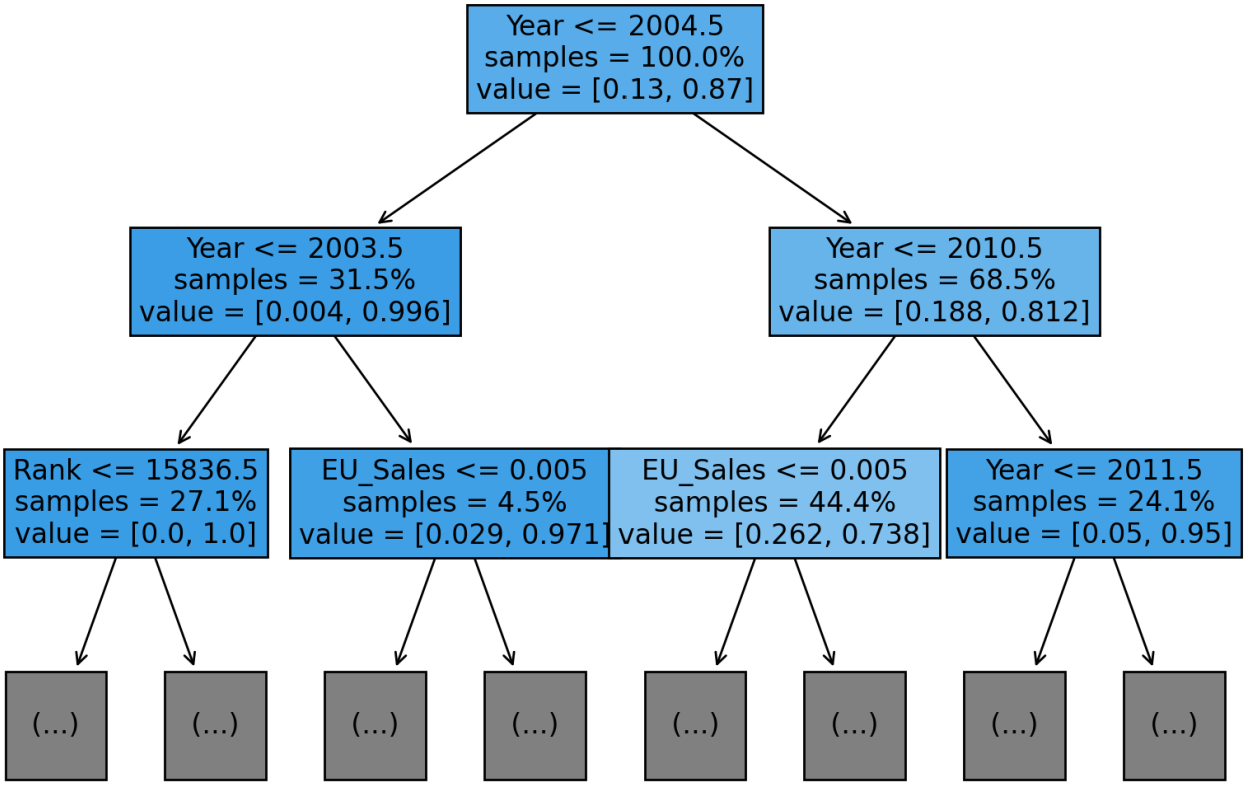

Рис. 6. Схема усеченного решающего дерева, полученного при решении задачи 4.

```
numerical_features = ['first_occurrence_date', 'last_occurrence_date',
                      'reported_date', 'geo_x', 'geo_y', 'geo_lat', 'geo_lon',
                      'victim_count', 'reported_time', 'reported_time_utc']
numerical_data = pd.DataFrame(index=data.index, columns=numerical_features)
numerical_data.loc[:, numerical_features] = data.loc[:,
    data.columns.isin(numerical_features)]

time_features = ['first_occurrence_date',
                 'last_occurrence_date', 'reported_date']

for col in time_features:
    numerical_data.loc[:, col] = numerical_data[col].apply(
        lambda timestr: pd.to_datetime(timestr, format='%m/%d/%Y %I:%M:%S %p'))
numerical_data['reported_time'] = \
    numerical_data['reported_date'].apply(lambda x: pd.to_datetime(
        x.strftime('%H:%M:%S')))
numerical_data.loc[:, 'reported_time_utc'] = \
    numerical_data['reported_time'].dt.hour * 60 * 60 + \
    numerical_data['reported_time'].dt.minute * 60 + \
    numerical_data['reported_time'].dt.second
numerical_data = numerical_data.astype('float64', errors='ignore')
# Отбросим значения даты и времени в формате Timestamp для нормализации
normalizible_features = ['geo_x', 'geo_y', 'geo_lat',
                         'geo_lon', 'victim_count', 'reported_time_utc']
normalizible_data = numerical_data.loc[:, numerical_data.columns.isin(
    normalizible_features)]
```

Код 46. Нормализация числовых данных, представленных в датасете.

```
geo_x                1.000000
geo_y                0.988746
geo_lat             -0.989791
geo_lon              0.999873
victim_count         0.000309
reported_time_utc   -0.004854
Name: geo_x, dtype: float64
geo_x                0.988746
geo_y                1.000000
geo_lat             -0.962731
geo_lon              0.988911
victim_count         0.001351
reported_time_utc   -0.003204
Name: geo_y, dtype: float64
geo_x               -0.989791
geo_y               -0.962731
geo_lat              1.000000
geo_lon             -0.987968
victim_count         0.000482
reported_time_utc    0.005366
Name: geo_lat, dtype: float64
geo_x                0.999873
geo_y                0.988911
geo_lat             -0.987968
geo_lon              1.000000
victim_count         0.000331
reported_time_utc   -0.004929
Name: geo_lon, dtype: float64
geo_x                0.000309
geo_y                0.001351
geo_lat              0.000482
geo_lon              0.000331
victim_count         1.000000
reported_time_utc   -0.003938
Name: victim_count, dtype: float64
geo_x               -0.004854
geo_y               -0.003204
geo_lat              0.005366
geo_lon             -0.004929
victim_count        -0.003938
reported_time_utc    1.000000
Name: reported_time_utc, dtype: float64
```

Код 47. Вывод кода 41 (Нахождение коррелирующих признаков, представленных в датасете).

```
clean_features = ['offence_code+extension', 'reported_time_utc',
                  'geo_x', 'geo_y', 'district_id', 'is_crime',
                  'is_traffic', 'victim_count']
clean_data = pd.DataFrame(columns=clean_features, index=data.index)

clean_data['offence_code+extension'] = data['offense_code'] * \
    10 + data['offense_code_extension']
clean_data['reported_time_utc'] = numerical_data['reported_time_utc']
clean_data['geo_x'] = numerical_data['geo_x']
clean_data['geo_y'] = numerical_data['geo_y']
clean_data['district_id'] = data['district_id']
clean_data['is_crime'] = data['is_crime']
clean_data['is_traffic'] = data['is_traffic']
clean_data['victim_count'] = data['victim_count']
clean_data = clean_data.dropna()
# print(f'Number of objects in cleaned dataset: {len(clean_data.index)}')

normalizible_clean_features = ['reported_time_utc', 'geo_x', 'geo_y',
                               'victim_count']
normalizible_clean_data = clean_data.loc[:, clean_data.columns.isin(
    normalizible_clean_features)]
std_scaler = preprocessing.StandardScaler()
normalized_clean_data = pd.DataFrame(
    std_scaler.fit_transform(normalizible_clean_data),
    columns=normalizible_clean_data.columns,
    index=normalizible_clean_data.index)

for col in normalizible_clean_features:
    clean_data[col] = normalized_clean_data.loc[:, col]
clean_data = clean_data.apply(
    lambda x: pd.to_numeric(x, errors='coerce')).dropna()

clean_data.to_csv('./clean_crime.csv')
```

Код 48. Подготовка данных к применению метода PCA.

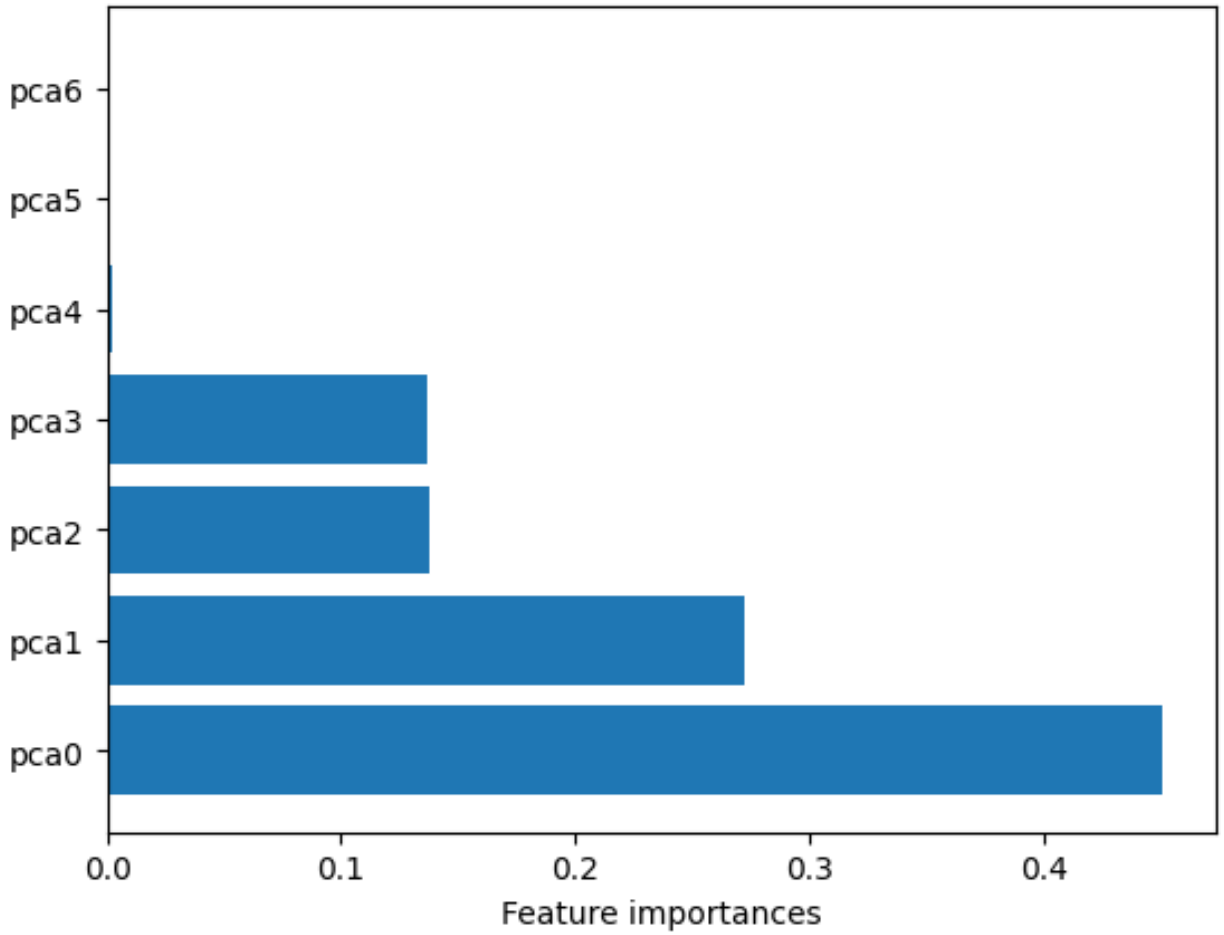

Рис. 7. График доли дисперсии объясненной каждой из компонент после применения метода PCA.

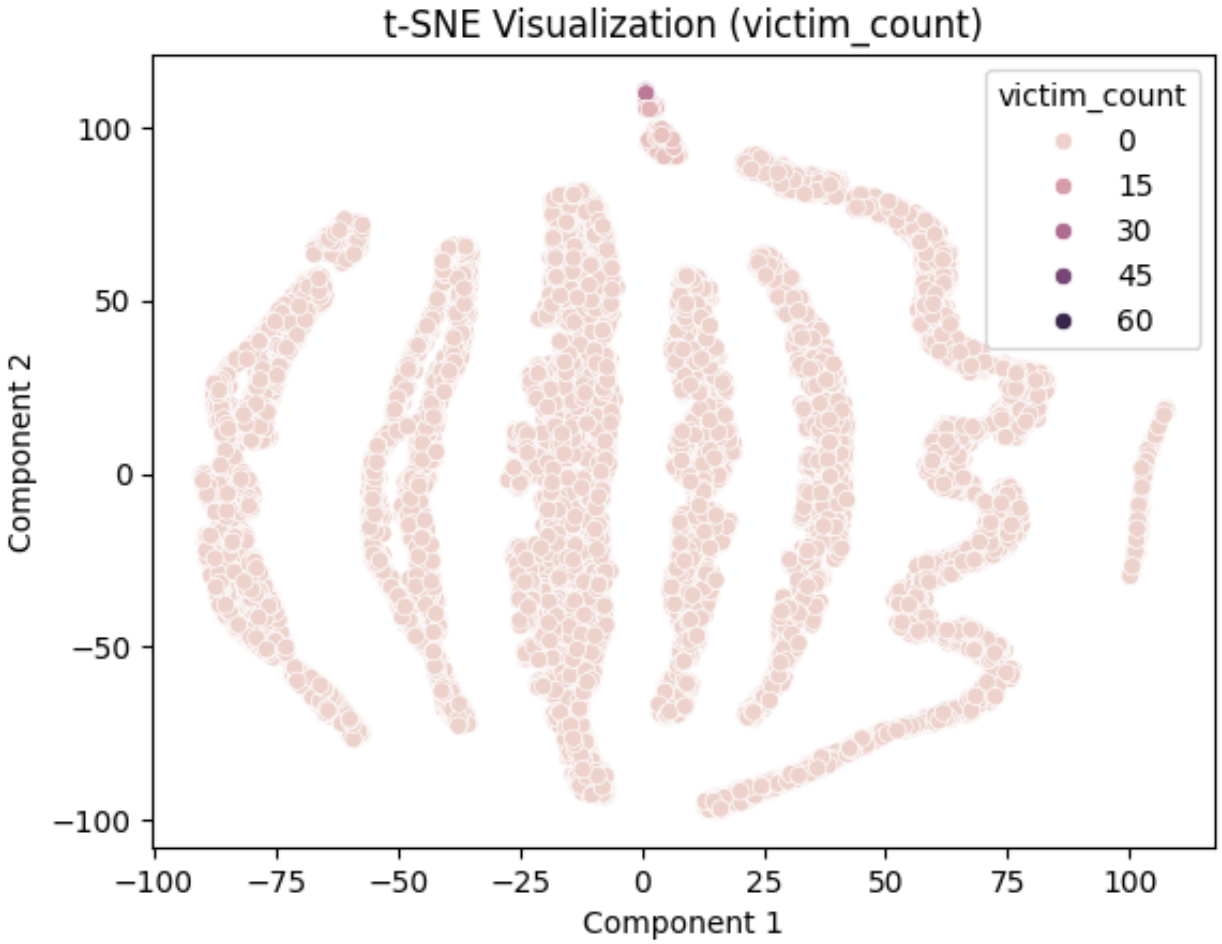

Рис. 8. Двумерная визуализация данных, полученная после применения алгоритма t-SNE (окраска по критерию *victim_count*).

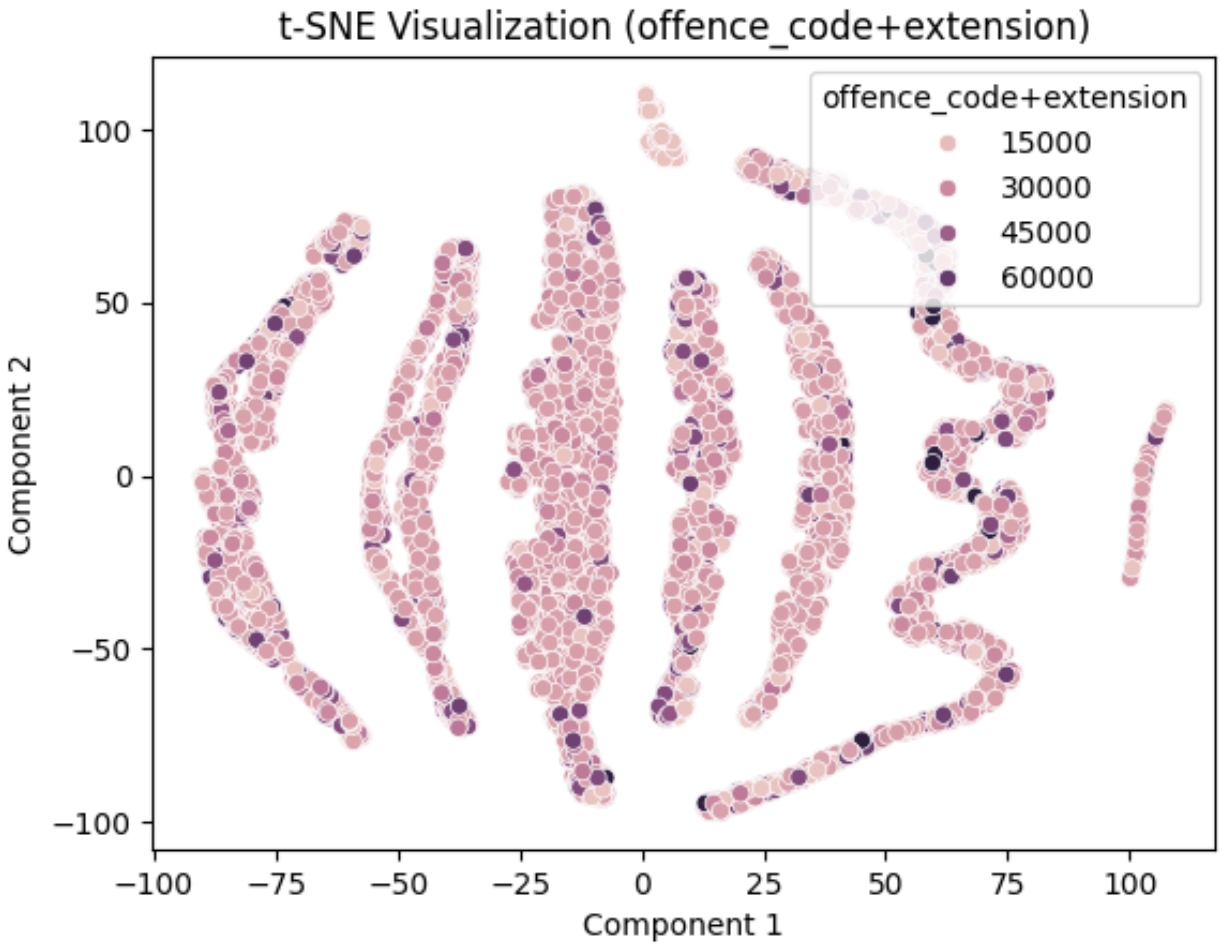

Рис. 9. Двумерная визуализация данных, полученная после применения алгоритма t-SNE (окраска по критерию *offence_code+extension*).